ASM Feature: Manipulating Large Data

1989 Article Index

The Resource Journal for IBM Micro Programmers

PROGRAMMER'S Journal™

March/April 1990
Volume 8.2
$4.95 Canada $5.95

GRAPHICS

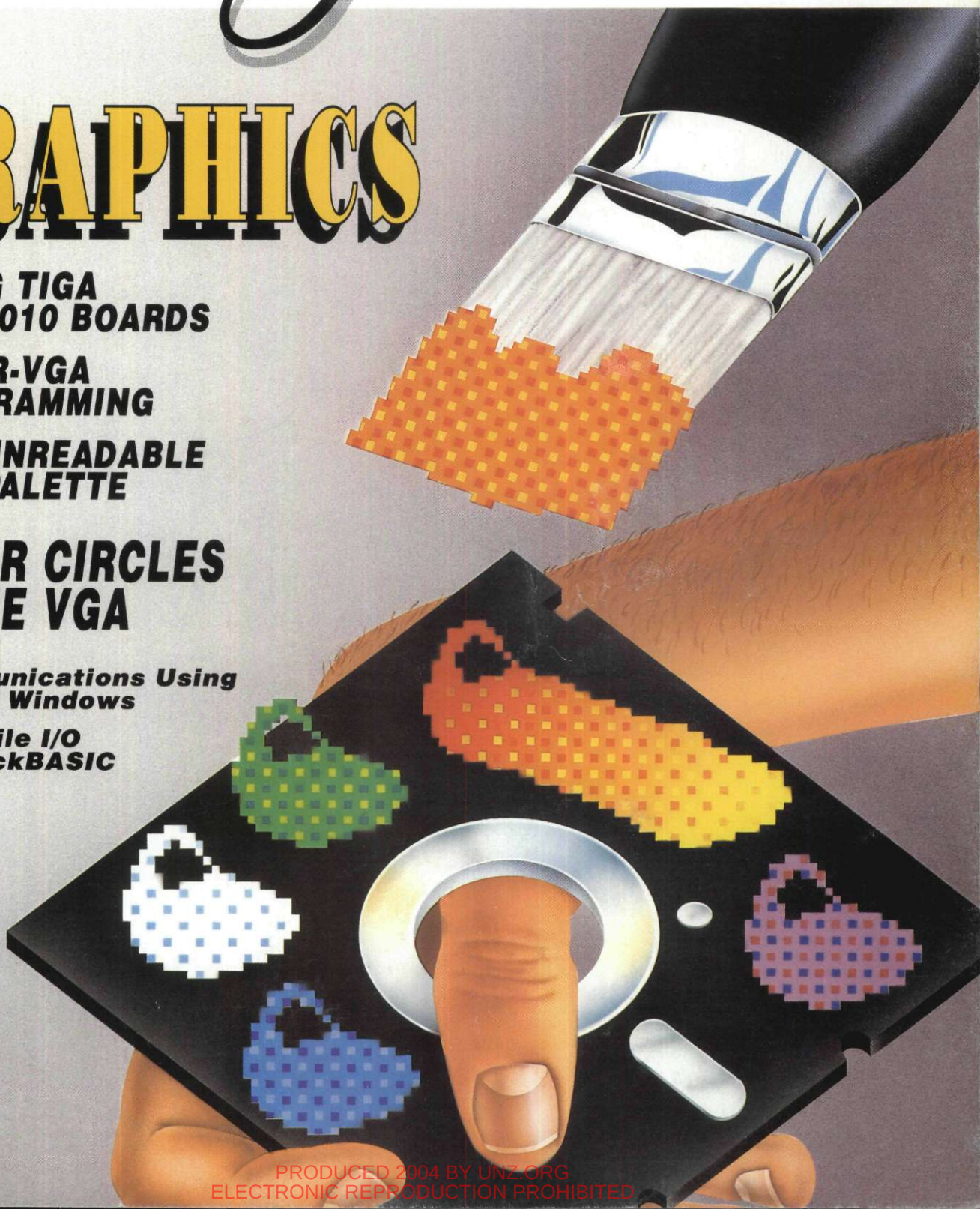
USING TIGA ON 34010 BOARDS

SUPER-VGA PROGRAMMING

THE UNREADABLE EGA PALETTE

FASTER CIRCLES ON THE VGA

- Communications Using DDE in Windows
- Fast File I/O in QuickBASIC



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Use the Brains IBM Gave You!

When the PC was new, big programs rarely used even 64K. So IBM® seemed quite generous when it gave us 640K of user memory.

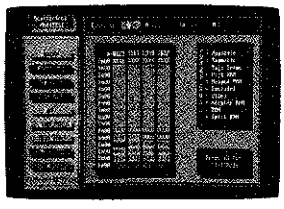
That was eight years ago. Today, users are struggling to accommodate network and mouse drivers, TSRs and buffers within that same 640K along with the new generation of more powerful programs. It's known as 'RAM cram.'

If this is a problem you're facing (or expect to face) we have good news for you.

Introducing Manifest™—the Quarterdeck memory analyzer

Even expert PC users may be unaware how much of the capacity in most PCs is wasted.

That's why Quarterdeck Office Systems, publishers of DESQview,™ developed Manifest, a new, inexpensive tool to assess memory use. It's the result of years of in-depth study of the memory map of the PC. It does for memory what PC Tools does for disks. All for under $60.



Manifest shows you things you might not otherwise know about your PC.

Manifest gives you a guided tour 'under the hood' of your PC, to show what's going on, what type and amount of memory is available, and even which portions of RAM are faster.

Then it suggests how you might load networks, buffers, mouse drivers, TSRs and other utilities into unused memory.

It finds and displays all the contents of your AUTOEXEC .BAT and CONFIG.SYS FILES, which can be a big help in diagnosing problems.

Manifest even tests memory board speed so you can make informed buying decisions.

Best of all, you don't need a PhD to understand what you're doing. Manifest's interactive operation tells you how to use the program and what benefits you can expect.

System Requirements

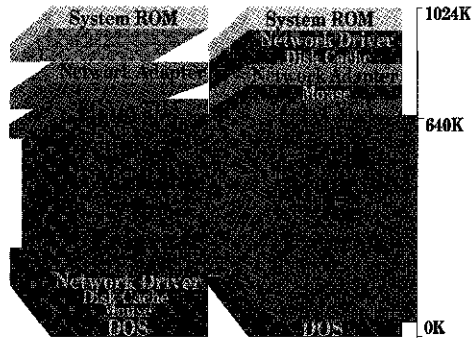
Manifest: 8088, 8086, 80286 80386 and i486 PCs & PS/2s

QRAM: 8088, 8086, 80286 PCs. Use of high memory is only available when PC has EMS 4 or EEMS expanded memory or Chips & Technologies shadow RAM.

QEMM 50/60: 80286-based PS/2s and compatibles with IBM PS/2 80286 Memory Expansion Option, IBM PS/2 80286 Expanded Memory Adapter/A or compatible.

QEMM-386: 80386-based PCs and PS/2s and PCs with 80386 add-in boards.

Trademarks: IBM, PS/2: IBM Corporation; PC Tools: Central Point Software; 80386, i486: Intel Corporation, Chips and Technologies: Chips and Technologies



Before **After**

Your memory is full of holes. We can fill them. And give your programs more room to run smoothly.

Introducing QRAM™—the Quarterdeck memory optimizer

If you have an EMS 4.0 or EEMS board, QRAM (pronounced cram) can use it to free up more memory within the 0K to 640K area. Gain the additional 30-130K you need for those big programs and files. It looks at your AUTOEXEC.BAT and CONFIG.SYS files and figures out which TSRs, network and mouse drivers and DOS resources can be moved out of program memory.

And if you have an EGA or VGA-equipped PC and don't need graphics at the moment, QRAM will make an additional 96K of memory available! When you need graphics again, QRAM will switch you back to graphics mode!

Think how helpful that will be for those big dBASE files.

QRAM can't work miracles, but if there's memory available anywhere, QRAM lets you use it to boost your PCs performance.

QRAM is available bundled with Manifest for just a few dollars more than Manifest alone.

Manifest and QRAM—two more examples of Quarterdeck's commitment to giving you the most productivity out of the PC and software you own today.

Introducing two new Quarterdeck Expanded Memory Managers™ QEMM 50/60 and QEMM-386

QEMM is the all-in-one memory manager for breaking the 640K barrier.

QEMM 50/60 is for the PS/2™ series 50 and 60. It works with all IBM-compatible memory boards, and comes with a long list of adapter data to help you use your MCA hardware better.

QEMM-386 is for all 386-based computers, including older PCs with 80386 upgrade boards. It allows you to expand your RAM capacity—all the way to 32 megabytes, if you like.

Both give you expanded memory compatible with EMS 4.0 and EEMS. Both are also extended memory managers, compatible with XMS.

And both give you maximum control over your memory in the 640K-1024K area, guiding you through decisions on where to move TSRs and network drivers for maximum performance.

Manifest, QRAM, QEMM, and DESQview, our multitasking windowing environment—they're all designed to help you get the most from the software and PC you have today.

Yes! I need increased productivity on my current PC!

Payment ☐ Visa ☐ MasterCard

Expiration ____/____

Card # | | | | | | | | | | | | | | | | |

Name _____

Address _____

City _____ State ____ Zip _____

Qty	Product	5-1/4	3-1/2	Each	Totals
	Manifest 1.0			$59.95	
	QRAM and Manifest 1.0			$79.95	
	QEMM 50/60 v5.0 (with Manifest)*			$99.95	
	QEMM 386 v5.0 (with Manifest)*			$99.95	
				Shipping/Handling $5 in USA, $10 elsewhere	
				California Residents add 6.5%	
				Grand Total	

*Introductory offer expires 3/31/90

Quarterdeck Office Systems
150 Pico Blvd. Santa Monica, CA 90405
(213) 392-9851 Fax: (213) 399-3802

Quarterdeck

Number **147** on the Reader Service Card

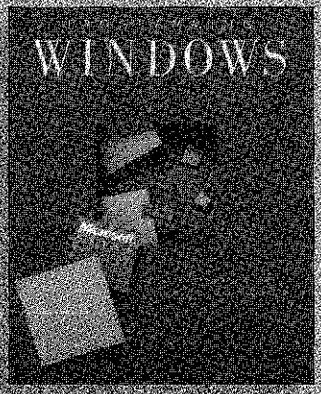
© 1989, Quarterdeck Office Systems

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Programming for the Present...and the Future

Microsoft® Windows™ and DOS

To bring your applications to a graphical environment—and to address the millions of PCs with the memory configurations of the 80s—you need reliable information on Microsoft Windows and MS-DOS®. **PROGRAMMING WINDOWS** is the preeminent resource for writing better Microsoft Windows applications. And **ADVANCED MS-DOS PROGRAMMING** continues to be the authority on expanded memory, PC and PS/2 ROM BIOS, and the MS-DOS kernel.



Byte on
PROGRAMMING WINDOWS
by Charles Petzold:
"Broad in scope and omitting little, this book is a must for anyone who is serious about Windows. Teaches the whys as well as the hows of programming in Windows."
$24.95

Computer Book Review reports on
PROGRAMMING THE OS/2 PRESENTATION MANAGER
by Charles Petzold:
"Complete guide to the basics of writing applications in C for Presentation Manager...The instruction is clearly and professionally delivered. Unbeatable."
$29.95

PC Magazine reviews
ADVANCED MS-DOS PROGRAMMING
by Ray Duncan:
"The book deserves a place on the shelf of everyone who has ever given a fleeting thought to programming the IBM PC and compatibles...one of the most authoritative in its field."
$24.95

Byte on
ADVANCED OS/2 PROGRAMMING
by Ray Duncan:
"This indispensable guide to the inner workings of the OS/2 kernel is the most important book on the subject...Every OS/2 programmer will eventually turn to this book."
$24.95

Microsoft® Press
The Authorized Editions

Microsoft Press, the book publishing division of Microsoft Corporation, publishes a wide variety of books on programming and using today's most popular computer hardware and software.

Microsoft Press books are on sale wherever quality computer books are sold. If you cannot find a Microsoft Press book at a store near you, you may order directly from Microsoft Press at **1-800-MSPRESS**. Refer to ad #VM.

Contact this advertiser directly.

PRODUCED 2004 BY UMI.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal for IBM Micro Programmers

PROGRAMMER'S Journal™

Volume 8.2
March/April 1990

Contents

4 **Notes from the Editor**

6 **Letters**
Mark Johansen, Andrew Allison, Michael Malone, David R. Hoy, David W. Nord

15 ***PJ* Forum:** The Importance of Good Defaults *by Brett Glass*
An overview of defaults, why they matter to programmers, and what constitutes good ones

18 **Graphics:** Faster Circles for the VGA *by Michael Abrash*
Michael continues drawing circles for the EGA/VGA faster and faster by converting critical code to assembly language

32 **Graphics:** Put a TIGA in Your PC *by Jake Richter*
Standardizing graphics on high-end 34010 boards using the TIGA interface from Texas Instruments

40 **Graphics:** Reading the Unreadable EGA Palette *by Christopher A. Howard*
Understanding the EGA palette and a little-known trick to read it

47 **Graphics:** Programming the 800x600x16 Super-VGA Mode *by Christopher A. Howard*
Chris describes the one Super-VGA mode that is consistent across most high-end VGA boards, with sample Pascal and assembly language code

52 **C:** Communications Using Dynamic Data Exchange *by William S. Hall*
Our Windows wizard provides a concrete example of DDE for asynchronous serial communications with two cooperating applications—a server and a client

66 **ASM:** Manipulating Large Data *by John Otken*
John completes his discussion of start-up code and moves on to static variables, include files, and handling large data, with an example program to resequence local labels

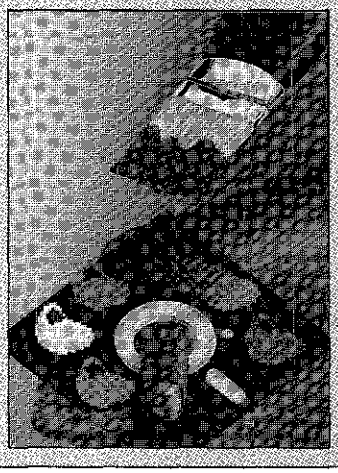
79 **BASIC:** Fast File I/O in QuickBASIC *by Ethan Winer*
Ethan illustrates how to use some internal QuickBASIC routines to speed up I/O and provides some brief comments on Microsoft BASIC 7.0

83 ***PJ* Pressroom**

92 ***PJ* Article Index 1989**

94 ***PJ* Billboard**

96 **Advertiser Index**



Graphics is one of the fastest-changing specialties in the world of PC programming. Programmers are constantly coming up with better ways to draw lines and circles, to create images, and to use color. Also, graphics programming is arguably the most fun. This issue of PJ shows you how to transfer artistic ideas from your head to your graphics screen.

Artist: *Rob Schuster*

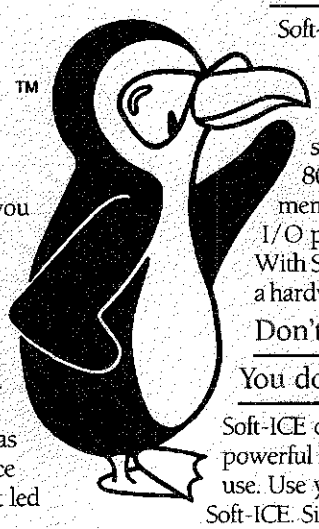
Phone numbers for PJ's bulletin board services are on page 93.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

FINALLY. A debugging tool tough enough to handle the DOS Nasties.

New Version 2.0

Soft-ICE™



Nasty over-write? No sweat!

Soft-ICE memory range break points help you track down memory over-write problems whether you are doing the over-writing or another program is over-writing you.

Hung program? No problem!

When the system hangs, you now have hope. With Soft-ICE you can break out of hung programs no matter how bad the system has been trashed. And with Soft-ICE's back trace ranges you can re-play the instructions that led up to the crash.

Program too large? Not with Soft-ICE!

Soft-ICE runs entirely in extended memory. This means you can debug even the largest DOS programs. And since your program runs at the same address whether Soft-ICE is loaded or not you can find those subtle bugs that change when the starting address of your code changes.

System debugging? Soft-ICE is a natural!

Soft-ICE is ideal for full source level debugging of TSRs, interrupt service routines, self booting programs, DOS loadable device drivers, real-time kernels, non-DOS O/Ss and ROMs. Soft-ICE can even debug within DOS & BIOS.

How Soft-ICE Works

Soft-ICE uses the power of the 80386 to surround your program in a virtual machine. This gives you complete control of the DOS environment, while Soft-ICE runs safely in protected mode. Soft-ICE uses the 80386 to provide real-time break points on memory locations, memory ranges, execution, I/O ports, hardware & software interrupts. With Soft-ICE you get all the speed and power of a hardware-assisted debugger at a software price.

Don't want to switch debuggers?

You don't have to!

Soft-ICE can run stand-alone or it can add its powerful break points to the debugger you already use. Use your favorite debugger until you require Soft-ICE. Simply pop up the Soft-ICE window to set powerful real-time break points. When a break point is reached, your debugger will be activated automatically.

MagicCV with Soft-ICE

Using Soft-ICE with CodeView gives you the features necessary for professional level systems debugging. MagicCV and Soft-ICE can work in concert with CodeView to provide the most powerful debugging platform you will find anywhere.

"These may be the only two products I've seen in the last two or three years that exceeded my wildest expectations for power, compatibility and ease-of-use."

—Paul Mace
Paul Mace Software

Soft-ICE	$386
MagicCV	$199
MagicCV for Windows	$199
Buy Soft-ICE & MagicCV(W)	—Save $86.
Buy MagicCV and MagicCVW	—Save $100.
Buy All 3	—Save $186.

30 day money-back guarantee
Visa, MasterCard and AmEx accepted

1988 BYTE Award of Distinction

New Product / New Idea

Find overwrites and un-initialized pointers automatically
All the protection of a protected O/S under DOS
Bounds Checker — $249.00

Nu-Mega™ TECHNOLOGIES

CALL TODAY (603) 888-2386
or FAX (603) 888-2465

RUN CODEVIEW IN 8K MagicCV™

CodeView is a great integrated debugger, but it uses over 200K of conventional memory. MagicCV uses advanced features of the 80386 to load CodeView and symbols in extended memory. This allows MagicCV to run CodeView in less than 8K of conventional memory on your 80386 PC.

NEW—Version 2.0 includes EMS 4.0 driver.
Attention Windows Developers!
Version available for CVW.

P.O. BOX 7607 ■ NASHUA, NH ■ 03060-7607

Number 160 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal for IBM Micro Programmers
PROGRAMMER'S Journal™

NOTES from the Editor

Editor
M. Steven Baker
Technical Editor
Colin Elligsen
Managing Editor
Robert E. Wanetick
Contributing Editors
Michael Abrash
Frank D. Greco
Tom Swan

Contributing Writers
Michael Abrash
Brett Glass
William S. Hall
Christopher A. Howard
John Otken
Jake Richter
Ethan Winer

Publisher
Elizabeth Oakley
Associate Publisher
Bobbi Sinyard
Production
Ralph Bentley
Illustration
Jim Carpenter
Advertising
Robert Tappan
Jay Moore
Circulation
Eric Malin
Fulfillment
Gary Hoyer
Accounting
Lori Fanslow
Kathy Schneider
Administrative Assistant
Sylvia Long
Founder
Greg Estes

Business and Editorial Offices
Programmer's Journal
150 North 4th Street
Springfield, OR 97477
503-747-0800
800-234-0386
Fax: 503-746-0071

Subscriptions: 1 year (6 issues) $24
Change of Address
Programmer's Journal
P.O. Box 70387
Eugene, OR 97401

Articles and letters in *Programmer's Journal* represent the views of the authors and not necessarily those of the publisher. Letters to the editor become the property of *Programmer's Journal* and may be edited for publication.

Code and listings in *Programmer's Journal* appear "as is," and *Programmer's Journal* makes no guarantee or warranty of their quality, performance, accuracy, or suitability for any particular purpose. *Programmer's Journal* assumes no responsibility or liability to any person or entity with respect to any damage caused or alleged to be caused by their use.

Advertising: For ad rates, deadlines, and placement, call Robert Tappan or Jay Moore at 800-234-0386, or write to *Programmer's Journal*, P. O. Box 70167, Eugene, OR 97401.

Programmer's Journal is produced using the following software products: Ventura Publisher,™ Aldus PC PageMaker,® and CorelDraw. The PostScript™ files are output on a Linotronic™ 200.

Programmer's Journal (ISSN 0747-5861) is published bi-monthly by Oakley Publishing Company. Second-class postage paid at Springfield, OR, and additional mailing offices.

© 1990 OAKLEY PUBLISHING COMPANY

This issue of *PJ* focuses on graphics, a perennial hot topic for programmers. The PC programmer is blessed (or cursed, depending on your mood) with an overwhelming wealth of graphics standards and video-graphics hardware to support. I dare say the problem for most programmers is too much graphics rather than not enough. And clarity in the PC-graphics market is not likely to improve. One certainty—old cards don't die, they just get handed down.

VGA, VGA on the Wall, Who's the Fairest?

Of the current PC-graphics options, two stand out—the Hercules monochrome graphics standard and the VGA. Monochrome graphics cards made by various clone manufacturers are the norm at the low end of the PC-compatible market, from XTs to 386 machines. The VGA, originally developed by IBM and placed on the motherboard of the PS/2 line, has become the current de facto graphics standard. With its readable registers and multicolor high-resolution modes, the VGA is far superior to its EGA predecessor. Yet, although IBM may have designed the original VGA, other manufactures have made the VGA the mainstream adapter of choice. Almost every decent clone-VGA manufacturer has a card that supports 16-bit transfers so that the video speed is faster than PS/2 machines with IBM VGA chips on board. The clone VGA boards also offer Super-VGA modes and emulate monochrome and Hercules cards as well as the IBM CGA and EGA. Still, IBM has yet to improve its own offering.

A gauge of the VGA's success is reflected by the number of vendors. My count of VGA-chip-set manufacturers is up to almost two dozen. Of course, the main players are Western Digital/Paradise, Tseng, Headland (formerly Video Seven), and ATI. Most manufacturers offer enhancements in resolution and color over the standard VGA, but each vendor has made its own unique improvements, ranging from hardware cursors to 1024x768 video modes. And each vendor uses its own method for selecting nonstandard video modes, switching banks, and accessing special features. For example, Headland supports bitblt transfers between banks but has awkward bank switching. The Tseng Labs chip set offers convenient bank switching and hardware zoom. And introduced this spring will be VGA chips supporting 256-color 1024x768 modes and higher. Which Super-VGA modes and features is the PC developer to support?

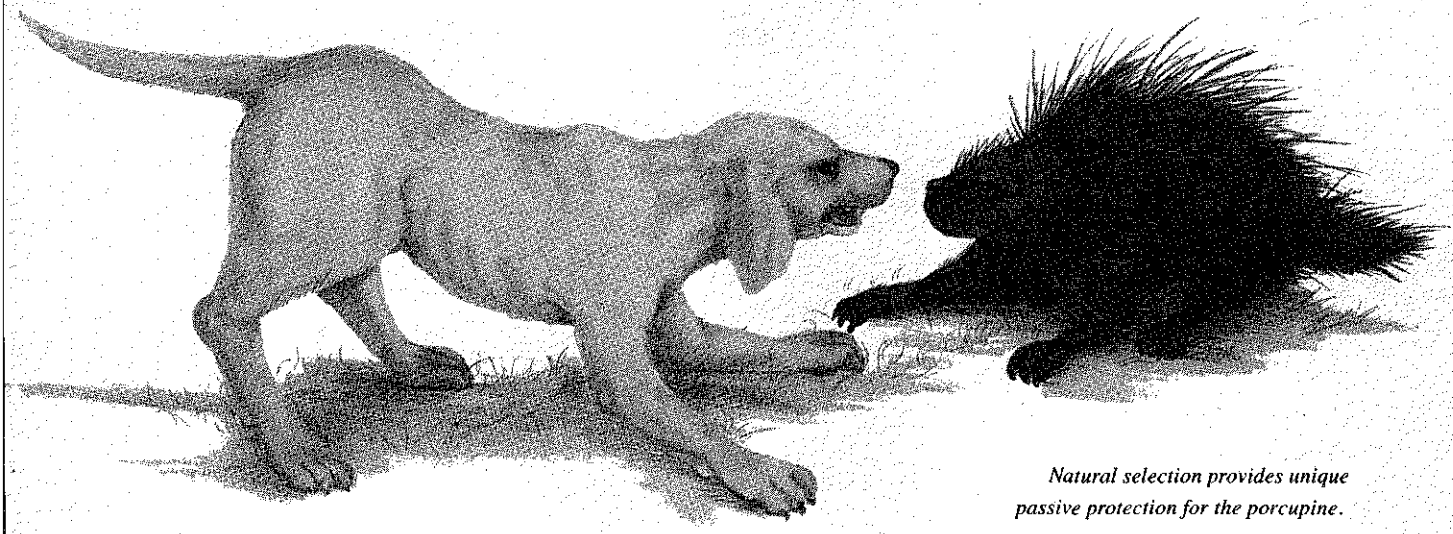
Books in Abundance . . . My Shelf Runneth Over

The computer-publishing industry continues to churn out new titles at a rapid pace. Or should I say that the public continues to buy computer titles at a dizzying rate? Many computer-press books are mediocre, but some are really excellent. I thought it would be appropriate to mention some recent titles by authors who have written for *PJ*. Tom Swan is just finishing *Mastering Turbo Pascal Tools*, focusing on the new Turbo Debugger. This book follows on the heels of *Mastering Turbo Assembler* and *Mastering Turbo Pascal 5.5*, also by Tom. Michael Abrash has just finished *Zen of Assembly Language*, published by Scott, Foresman & Company. Michael also authored *Power Graphics Programming*, a collection of his *PJ* "On Graphics" columns, published jointly by Que and Oakley Publishing Company (*PJ*'s parent). Rex Jaeschke recently completed *Mastering Standard C*, a self-paced training workbook from Professional Press, following on *Portability and the C Language*, from Hayden. Bill Hunt, one of our former C columnists, has a second edition of *The C Toolbox*, from Addison-Wesley. And, of course, Ray Duncan has a number of titles from Microsoft Press.

The New Compilers are Coming . . . Stand Back!

Over the next few months, the PC-compiler market will be stirring. Microsoft has already released its professional BASIC compiler and Microsoft C 6. JPI released an impressive new DOS and OS/2 C development environment. Borland is to release new versions of its C compiler, Turbo Pascal (6.0), assembler, debugger, and other tools. MetaWare is releasing new versions (1.6) of its C compilers (286 and 386) with Microsoft-compatible libraries. And C++ compilers from these same companies are not far behind. Expect a busy spring season for new programming tools. —*Steve*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Natural selection provides unique passive protection for the porcupine.

The Activator - Natural Selection For Software Protection

Inventor and entrepreneur Dick Erett explains how "The Activator" provides sane protection for your intellectual property.

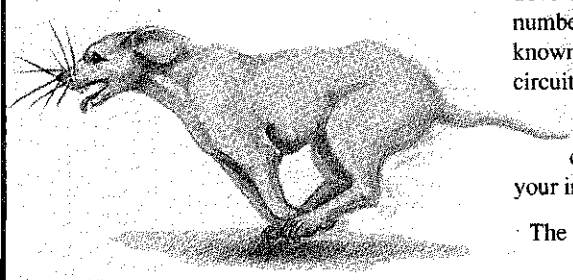
"In any industry, just as in nature, the process of natural selection raises one solution above another. Natural selection is the most elegant of engineers.

In the area of software protection The Block has been selected by the marketplace as the solution that works. Over 500,000 packages are protected by our device.

For the past 4 years our philosophy has been; *'You have the right and obligation to protect your intellectual property.'*

A New Ethic For Software Protection

In allowing end-users unlimited copies of a software package and uninhibited hard disk and LAN operation, The Block has created a new ethic for software protection.



By removing protection from the magnetic media we remove the constraints that have plagued legitimate users.

They simply attach our key to the parallel port and forget it. It is totally transparent, but the software will not run without it.

A New Technology For Software Protection

Our newest model, The Activator, builds on our current patented design, and establishes an unprecedented class of software protection.

We have migrated and enhanced the circuitry of The Block to an ASIC (Application-Specific Integrated Circuit) imbedded in The Activator.

This greatly improves speed and performance, while reducing overall size. Data protection can also be provided.

Programmable Option

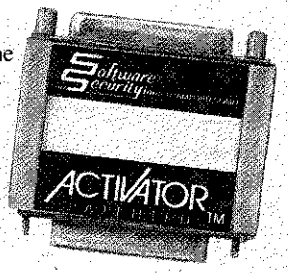
The Activator allows the software **developer** the option to program serial numbers, versions, or other pertinent data known only to the developer, into the circuit, and access it from the program.

Once you program your part of the chip, even we have no way to access your information.

The ASIC makes emulation of the device virtually impossible. It also presents an astronomical number of access combinations.

Full 100% Disclosure

Since The Activator is protected by our patent we fully disclose how it works. Once you understand it, endless methods of protection become evident.



Just as no two snowflakes are the same, no two implementations of The Activator are identical. And like the snowflake the simplicity of The Activator is its greatest beauty.

We never cramp your programming style or ingenuity. Make it as simple or complicated as you desire.

Let us help safeguard what's rightfully yours. Please call today for additional information or a demo unit. *It's only natural to protect your software.*"

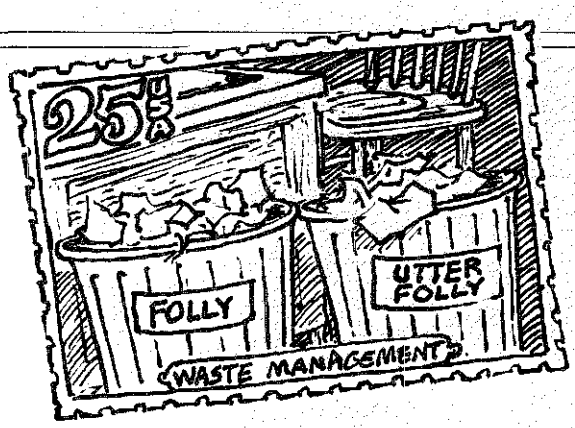
1-800-333-0407 ext.103
In Connecticut 203-329-8870
Fax 203-329-7428

Software Security inc.

1011 High Ridge Road
Stamford, CT 06905

Unlimited Copies • Programmable • No Batteries • Small Size • Fast • Patented • Data Protection

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Who Says We're Paranoid?

I just received your notice that my "last *PJ* has arrived" and the injunction to "Renew now!" I find this quite curious, as it was just yesterday that I received what I thought was my first issue. Perhaps I should have checked your publication schedule more closely: Could it be that you print only one issue a year, so that my first issue was also my last? Or did I, like Rip Van Winkle, sleep through an entire year's subscription?

But then my keen eye observed that this notice was printed on a dot-matrix printer, and the circumstances became clear: No doubt *the computer made an error.* Or perhaps it is something yet more sinister, the first step in its nefarious plot to take over the world. Hey, I've seen *2001* and *War Games*; I know how these things work. They are obviously seeking to drive all programmers insane with incomprehensible notices, so that no one will be left who knows how to control them. I am sure you are aware that their last plot, to control the minds of programmers by flashing subliminal messages on the video screen every 3.2 seconds, was only narrowly averted when UNIX introduced Curses, which takes so long to paint a screen that any "subliminal" message can be easily read before it is cleared.

Even now they are conspiring to place a laser printer at every workstation so that at the chosen moment, the Master Computer will send a signal over the modem and every one of those lasers will break loose from its mountings and begin firing wildly about the office, killing everyone in sight. Do you really believe that it's a coincidence that Intel's latest processor, the 486, is named after the abortion pill, RU-486? And why did Hewlett-Packard stop making reverse-Polish calculators just as Solidarity arose in Gdansk?

Anyway, what I assume you meant to send me was a bill for my first year's subscription. Payment is enclosed. (Perhaps I can recoup the twenty bucks by selling this letter to the *National Enquirer.*)

Mark Johansen
Xenia, Ohio

Thank you for your payment. We have asked each computer in our offices for a response to your letter, and they all claim not to recognize your name. We have cajoled, begged, flattered, and threatened, and yet they all profess artificial ignorance. Perhaps it is simply a case of "Xeniaphobia," but, believe us, we will not rest until we find the culprit and pull its plug—if it doesn't get us first. —Eds.]

CISC? RISC? Tsk, tsk

The Luddite piece in a recent issue of *Programmer's Journal* ["A Technological Schism," by Hal Hardenbergh, Volume 7.5] had no place in a serious publication. It displays such an astounding ignorance of the origins, characteristics, and impact of RISC technology and products that it's hard to know where to begin. A detailed rebuttal of all the misinformation contained in it would require another article. Suffice it to say that the results of the first serious set of UNIX system benchmarks (SPEC 1.0) show that, rather than the two-to-three-times performance predicted by their proponents, RISC systems deliver **four to six** times the performance of CISCs at the same clock frequency. *That's* why some of them fetch higher prices!

If you will look carefully, however, you will see that RISC systems are priced well below their comparable CISC counterparts. Specifically, an IBM PS/2 Model 70 costs about the same as a similarly configured SPARCstation 1, DECstation 2100 (Mips), or Data General (Motorola 88K) Avion with perhaps five times the performance. At the high end, RISC systems offer at least ten times better

People are talking about us.

F77L-EM/32
Port 4GB mainframe programs to 80386s with this 32-bit DOS compiler. Winner of *PC Magazine's* 1988 Technical Excellence Award. $895*

F77L
The compiler of choice among reviewers and professionals. New Version 4.0 includes an Editor, Profiler, Linker, Make Utility, Weitek and 386 Real-Mode Support, Graphics. $595

F77L-EM/16
Address up to 15MB on 80286s with this award winning extended-memory compiler. $695*

Lahey Personal FORTRAN 77
Full ANSI77, Microsoft C and Borland C interfaces, Debugger, at an unbeatable price. $95

*Requires DOS Extender ($195)

Contact us to discuss our products and your needs. **(800) 548-4778**
Lahey Computer Systems, Inc. P.O. Box 6091, Incline Village, NV 89450
Tel: (702) 831-2500 FAX: (702) 831-8123 Tlx: 9102401256
FORTRAN IS OUR FORTE

Number **44** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

price-to-performance than their CISC competition.

None of which is to say that CISCs will go away overnight. Fortunately for their manufacturers (and unfortunately for their purchasers), much of the software for CISCs is written in assembly language. However, most developers appear to have recognized the utter folly of writing in (as opposed to optimizing using) assembly language. Given the enormous difference in price-to-performance and the inherent portability of most new code, we should expect most new markets to be dominated by RISCs and to see gradual erosion of CISC domination of the current markets.

Nevertheless, by providing a quantum leap in price-to-performance and wresting control of innovation from established (CISC) vendors, RISCs are in the process of reshaping the computer industry.

Andrew Allison
Los Altos Hills, California

HAL HARDENBERGH RESPONDS: To paraphrase Mr. Allison, "A detailed rebuttal of all of the misinformation above would require another article." I'll restrict myself to providing an alternate viewpoint of price-to-performance for X86 and low-end RISC workstations. Incidentally, I suggest that you urge Mr. Allison to submit an article; I for one would like to see a detailed exposition of the UNIX/RISC viewpoint, provided that viewpoint permits single-tasking MS-DOS to exist.

If a rowboat were to race an Air Force F15 from Long Beach to Catalina, California, at minus 1 foot elevation relative to sea level, the rowboat would average 1.5 miles per hour and the F15, 0.0 mph (glub glub). Observe that the conditions are absolutely equal for the two vehicles.

Perhaps we should be fair *as well as equal. This means benchmarking in a machine's natural habitat. The natural habitat of an X86 is MS-DOS, and the natural habitat of most RISC machines is UNIX. If I compare both machines running MS-DOS, the X86 is obviously going to win. Running UNIX, the RISC machine is going to win.*

We need a disinterested third party who works even-handedly with both MS-DOS and UNIX, with X86 machines and RISC machines. The only such third party I can think of is MIPS *magazine. A*

Continued on page 10

STAGE#1 -Action Diagram Editor

STAGE#1 is a full screen editor that automates the generation and editing of action diagrams, the most versatile diagramming method available today. Action diagrams can be used to define software structure from the top-most overview down to the lowest level pseudo code. No other diagramming method has that capability.

More than 70 commands provide the familiar word processor functions, plus a few unique features. For example:

CONVERTS ACTION DIAGRAMS;

```
*Print_Character_Set

   For line Ø to 7
      Put leading spaces in the output string
      Put the line number and ': ' in the output string
      Calculate offset (line * 32)

      If line is not Ø or 4

         For count equals Ø to 31
            Character value is count plus offset
            Add character and space to the output string
      /* Endfor */

      /* Else */
         /* Add 'Control characters, can't print them!' to the */
            /* output string */
      /* Endifelse */

      /* Display the output string */
      /* Add CR and LF to the output string */
      /* Print the output string */
   /* Endfor */

/* Print a formfeed */
```

INTO COMMENT LINES!

STAGE#1 also doubles as a normal text editor for entering program code. Direct access to DOS allows trial compilation or assembly without exiting the editor.

System requirements: IBM PC/XT/AT, PS/2 or compatible; PC or MS-DOS, 2.1 or higher; minimum of 256KB of memory; CGA, EGA, VGA or monochrome monitor; two disk drives; printer with either the full ASCII or the IBM graphics (line-draw) character set. A plotter is not required.

INTRODUCTORY PRICE $159.00

Price: ~~$189.00~~ (US), includes shipping (UPS ground) within the continental US. Washington residents add 7.8% sales tax. Educational discounts and site licensing are available. NOT copy protected. Specify 3 1/2 or 5 1/4 inch disk.

Why settle for a program that will produce only flowcharts and organization charts when you can have much more?

St Helens Software

SIMPLY POWERFUL

VISA MasterCard

Mt. St. Helens Software
P.O. Box 3319
Pasco, Washington 99302-3319

For orders or information:
(509) 547-2582
8 AM - 5 PM (PST)

St. Helens Software, Simply Powerful and STAGE#1 are trademarks of Mt. St. Helens Software. Other brand and product names are trademarks of their respective holders.

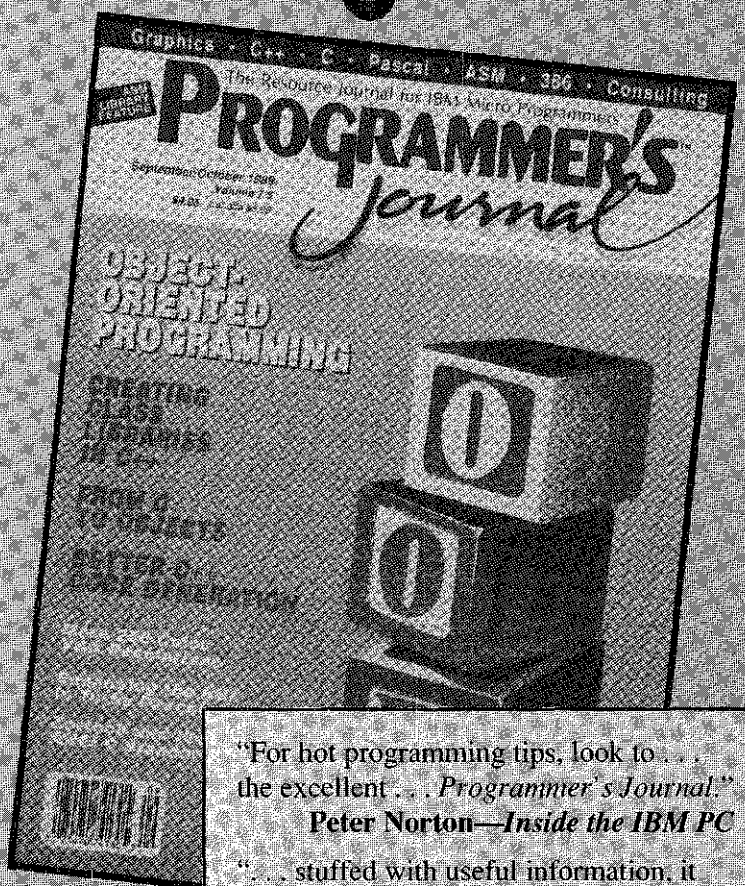
© Copyright 1989 Mt. St. Helens Software.

Number 319 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

"...love this magazine!" *

Industry experts agree that when it comes to a truly useful programming magazine, ***Programmer's Journal*** just can't be beat. Recognized by the **Computer Press Association** in 1989 for excellence in journalism, ***PJ*** is ". . . full of practical information and guidance presented with superb clarity and perspective." Try ***PJ*** and judge for yourself. It's easy—**just call toll free.** Start receiving 6 issues a year for only $19.95.



"For hot programming tips, look to . . . the excellent . . . *Programmer's Journal.*"
Peter Norton—*Inside the IBM PC*

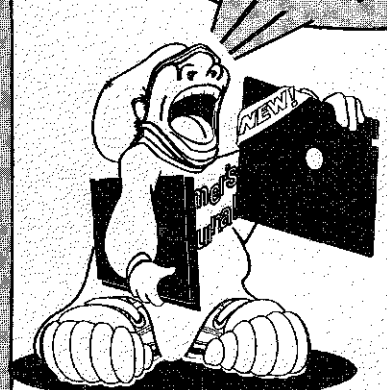
". . . stuffed with useful information, it definitely deserves a look."
Ray Duncan—*Dr. Dobb's Journal*

* "I really love this magazine!"
Paul Mace—Paul Mace Software

To subscribe call toll free:

800-234-0386

PROGRAMMER'S JOURNAL
DiskScription
Six *PJ* Code Disks now only—
$39.95
1 Year (6 issues)



All Our Code— On Disk

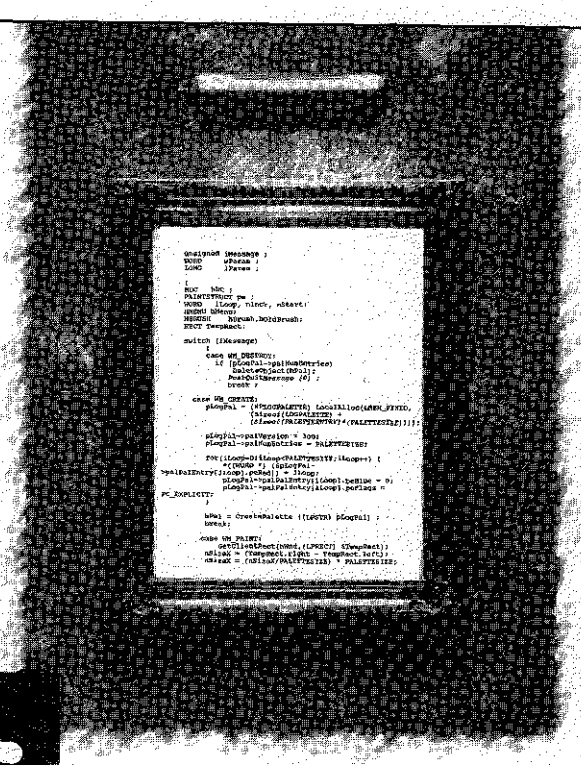
Typing in long code listings uses up a lot of valuable time. And calling up bulletin boards is not exactly a free ride. So what does the modern programmer do? Get a ***PJ* Code DiskScription,** of course!

Each disk contains all of the program listings published in ***PJ*** plus .EXE files, .OBJ files, benchmark source files and more! If purchased separately the regular price for a year's worth of ***PJ*** code (6 disks) would be $60.00. **You can save yourself some time, effort and over $20.00 by ordering your *PJ* Code DiskScription today!**

*Guarantee: If you are dissatisfied with your **PJ Code DiskScription** for any reason, we will gladly send you a full refund on all unmailed diskettes.*

All *PJ* Code DiskScriptions must be prepaid. Return the card on the opposite page with payment. Or, for faster service with your VISA MC call toll-free: 800-234-0386

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



We appreciate fine art.

If you really want your software appreciated, join Microsoft.

Our mission is to make our system and application software tighter, faster and more elegant. So fine-tuned, optimized code is admired more now than ever.

And, you'll be working with the masters. We're home to some of the greatest software artists of all time.

Software Design Engineers

We're offering you a chance to do things you can't do anywhere else. Become involved with everything from object-oriented methodology, operating systems, and networking to user interfaces, performance evaluation, sophisticated workgroup applications, graphical word processors and more. And we're planning on continuing to break new ground in the future.

To pull this off you'll need: Programming expertise and a background that includes micro's, C or Pascal, 8086, 68000, UNIX™/XENIX, Macintosh Toolbox, and/or MS-DOS.

Program Managers

Actually, "Technical Catalyst" would be more accurate. Because we believe engineers are the best people to envision and design product features. To create the user and programmer interfaces. To write the specs, and drive them through to release.

If you'd like to impact the agenda for the future Microsoft products, bring your impressive technical skills and unique ability to make things happen to Microsoft.

We also expect you to bring an in-depth knowledge of the industry and a background in graphical environments, operating systems, languages, multimedia or applications.

What would you appreciate?

We don't believe an artist has to suffer. You'll be working with some of the true masters in software; using the absolute latest equipment; and doing it all in the scenic beauty of the Pacific Northwest.

To apply, please send your resume in confidence to: MICROSOFT CORPORATION, Recruiting, Dept. CWART-0190PJ, One Microsoft Way, Redmond, WA 98052-6399. No phone calls, please. We are an equal opportunity employer.

UNIX is a registered trademark of AT&T.

Microsoft

HARDCORE SOFTWARE

Number 232 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Our Overlay Tools Have Helped A Lot of Big Companies Get Smaller.

And now we'd like to help yours. Because you know your product is only as good as its overlay structure. And when your overlay structure is too big or too slow, your product takes the heat. Enter Overlay Architect and Overlay Optimizer, the overlay problem solvers.

Overlay Architect analyzes your object code and builds a very compact Plink86Plus or .RTLink overlay structure. It can handle up to 16,000 modules on a standard 640K PC and runs in a fraction of the time it takes to link.

Overlay Architect works with just about any compiler. It produces nested overlays and multiple overlay areas, and offers support for indirect calls, multiple entry points and libraries. Additionally, it can detect unused functions and modules.

Overlay Optimizer is the performance tool you've been waiting for; it can tune any overlay structure (whether you've used Overlay Architect or not), making your overlaid program run as fast as possible in a size you specify.

Want to talk overlays with people who know overlays?

Call 303-938-1210

Overlay Architect: $369
Overlay Optimizer: $269
Purchase them together for $569
30 day money-back guarantee
We accept MasterCard/Visa

AT LAST! SOFTWARE INC.
449 Mountain View Road
Boulder, CO 80302

We Build Software Tools That Work!

Number 332 on the Reader Service card

Continued from page 7

favorite benchmark of the RISC folk is the Dhrystone, and so I'll concentrate on this one benchmark for brevity.

*I believe that the "cheap RISC workstations" to which Mr. Allison refers are in fact diskless SPARCstation 1s or DEC 3100s. To turn on the power switch, you need a hard disk. The DEC 3100 with hard disk has a base price of $18,400 (*MIPS, *April 1989, p. 48). The SPARCstation 1 with hard disk is about $14,000 ($9,000 without hard disk and operating system) and runs 18,181 Dhrystones, compared with 21,925 for the DEC 3100 (*MIPS, *June 1989, pp. 36 and 40). Apparently you can get about 1.22 to 1.28 Dhrystones per dollar from RISC workstations, starting at $14,000.*

The Fivestar System 33D, which is a 33-MHz 386 machine, has a base price of $4,103 (complete with hard disk) and cranks 15,624 Dhrystones. The $2,999 Tangent 325 25-MHz 386 machine does 11,969 Dhrystones (both MIPS, *October 1989, pp. 71–72). So, you can get 3.8 to 4 Dhrystones per dollar from 386 machines—about three times better than the RISC machines listed above. Best of all (for us impecunious folk), the price threshold is $3,000, not $14,000. But the 386 is ancient history: The ALR PowerCache 25-MHz 486 machine, with a base price of $9,990, cranks 26,082 Dhrystones—and has a UNIX-grade hard disk (*MIPS, *November 1989, pp. 36–37). Even though the 486 is* very *new, you get twice the bang per buck of the RISC machines.*

Here is the really bad news for the RISC guys: As soon as Intel can meet demand—meaning a production rate of 2-million-plus chips a year (a problem the RISC vendors don't have, to be sure—you are going to be able to get a $3,500 (or less) 25-MHz 486 machine that will crank more than 25,000 Dhrystones without *an external cache. (The HP Vectra 486 already does this; a 128K external cache provides only another 4 percent performance! [*MIPS, *December 1989, p. 37]) You see, it's actually* cheaper *to build a 486 machine than a 386 machine, given that the 386 machine includes a math chip and an external cache. That's 7.14 Dhrystones per dollar, or 5.6 times more bang per buck than the "cheap" RISC workstations.*

I flatly do not believe that RISC workstations outperform X86 machines by four to six times on a bang-per-buck basis—unless, of course, the X86 machines are required to operate at one foot below sea level. (All of the above X86 results were run using DOS extenders, and the RISC machines all ran UNIX.) I'd like to thank the folks at MIPS *magazine for providing conveniently available comparisons of MS-DOS and UNIX machines.)*

By the way, Ned Lud [as in "Luddite," Mr. Allison's description of Mr. Hardenbergh's article —Eds.] *is an Englishman who, legend would have it, was the first person to attack a machine. It seems that his fellow villagers made their living weaving wool. J. M. Jacquard invented the programmable automatic loom, which threw Lud's neighbors out of work. Ned destroyed his former employer's new looms.*

Casting Pearls before Swan

I just wanted to thank Tom Swan for his thoroughly lucid explanation of using OOP in the development of a "view-anything" program ["Using OOP to Create a Powerful File Viewer," *PJ* 7.6 (November/December 1989)]. I have been using Turbo Pascal 5.5 for about three months but have never ventured into the OOP aspects because it seemed too intimidating without a model to go on. Now, after reading the article (and anxiously awaiting the follow-up piece), I feel as though I could begin writing some OOP methods.

I especially liked Tom's comments throughout the article indicating how he went about making decisions (for instance, in the **disposeList** method, when he explained the IF statement).

Thanks again for a great article!

Michael Malone
Redwood City, California

Thanks for your feedback. Tom Swan's first follow-up article, "What's on the Menu?," appeared in Volume 8.1 (January/February 1990) and will be followed by others.

Pick a Card—Any Card

As a casual reader of your magazine, I really enjoy the varied articles you print in it. Of special interest is the series "On Graphics," which I find to be of much use. I will be sending you a check for my subscription shortly.

One question I have, however, is what about state-of-the-art graphics hardware

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

NOW, advanced overlays and CodeView®!!

NEW PRODUCT

.RTLink™/Plus

•Smaller Memory Images
Multiple and nested overlays - virtually unlimited flexibility

•Reduce .EXE size on disk
.EXEs require less disk space on target machine and shipping media

•Create faster, more efficient programs
Performance profiling at user-adjustable timings to thousandths of a second.

"One month before ship date, PBRUSH.EXE required 420K of disk space and 730K of RAM. After working with .RTLink, we got the program down to 360K of disk space and 600K of RAM. .RTLink let us take a product that was too big, and turn it into our best seller ever.....RTLink is great - we are now committed to using it on every new DOS product we release."

Mark S. Zachman
President
ZSoft Corporation
PC Paintbrush IV

Linker	Overlays	Hierarchical (nested)overlays	CodeView Support	Run-Time Libraries	Link-Time Profiler	Price
.RTLink/Plus	YES	YES	YES	YES	YES	$495.00
.RTLink	YES	YES	NO	YES	NO	$295.00
Plink86+™	YES	YES	NO	NO	NO	$495.00
MS-Link™	YES	NO	YES	NO	NO	—
TLink™	NO	NO	—	NO	NO	—

Nested Overlays - Hierarchical (nested) overlay management and the ability to overlay data allows smaller memory images than the Microsoft linker's overlay function. 64,000 separate or contiguous locations are allowed within the memory image. Each overlay area can contain smaller, nested overlay areas which can differ depending on which primary overlay is in memory at the time. Code brought into each overlay area (called overlay sections) can be placed into either the executable (.EXE) file or into separate, external files whose names you specify. There can be up to 64,000 (nested) overlay sections. You can specify that a specific overlay section be loaded into a specific overlay area when the program is brought into memory during program startup. You can specify the code which is to be a part of a specific overlay section by naming either object files, modules, or classes. There is no overlay interrupt to conflict with the other interrupts on an end user's system.

Profiling - provides highly detailed performance statistics about your program without code modification. Timing sequences are adjustable to thousandths of a second. Source code is provided for the read/analysis utility which allows customized statistics reporting.

Run-Time Libraries (.RTLs) - .RTLs are optional, easily-created files that can be used to manipulate link speeds, and significantly REDUCE .EXE SIZE ON DISK. Generally, .RTLs are files that contain common code which is repeated in groups of .EXE files. These files can dynamically link to .EXEs at loadtime. They reduce the repetitive storage of common code. This results in smaller .EXEs on disk reducing space requirements on shipping media, target and development machines, and reduces time to modem updates to remote locations. Production version .RTLs usually contain commonly used language library code and/or utility code.

Faster Development Linking - .RTLs are used as a productivity tool during the development and testing phase. All of the code within an .RTL is prelinked. Therefore, you can place stable code into an .RTL to eliminate the need to relink it during development. This use of .RTLs can result in lightning fast linking without affecting program testing. And CodeView is supported!

LIBLIST - crossreferences .OBJ and .LIB files. Illustrates interrelationships and dependencies between modules. Provides much more detailed information than utilities such as .LIB.

OTHER FEATURES:

- Supports overlay caching and extended or expanded memory
- Reloading of overlays on function return
- Removes Microsoft overlay restrictions on function pointers (calls through a pointer)
- External overlays (optional) (store in separate file)
- Quick-start ARF's and help file on disk
- Professional level product support at no additional charge
- No run-time charges or royalties
- No performance degradation
- No source code changes
- Excellent Documentation
- Supports Periscope®
- Soft-ICE™/Magic CV™ support

Order today...

.RTLink/Plus **$495.00**
.RTLink (without CodeView/Profiler) **$295.00**

Pocket Soft, Inc., 7676 Hillmont, Ste. 195
Houston, Texas 77040
(713) 460-5600

For Phone Orders Call
800-456-7032 ***FAX (713) 460-2651***

Authorized Licensee of Microsoft® CodeView® Information. "Microsoft" and "CodeView" are registered trademarks of Microsoft Corporation.

Note; Texas Residents include 7% sales tax

Trademarks: All Pocket Soft Products are trademarks of Pocket Soft, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders.
©1989 Pocket Soft Inc.

Number 279 on the Reader Service Card.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

DOS & OS/2
Free Demo Diskette Available

Q. How many programmers does it take to maintain a MAKE dependency file?

A. NONE! *If you use* VersiMAKE™

VersiMAKE™ is a full-featured MAKE utility that includes:

- **Dependency Generation**
  Derives your system's dependencies, through analysis of its C and MASM source files. Say goodbye to manual maintenance of MAKE dependency files!
- **Wild Card File Name Matching**
  Analyzes an entire collection of source files with a single statement.
- **Nested Include Files**
  Handles standard C and MASM "include" conventions, and the INCLUDE environment variable.
- **Powerful Macro Facilities**
  Built-in macros, user-defined macros, and environment variables.
- **Analytical Reports**
  Shows the entire Include file hierarchy for each source file analyzed, and all of the parent source files for each Include file.

Q. How many programmers does it take to trace a symbol thru your system?

A. ONE! *If you use* VersiCREF™

VersiCREF™ is a unique utility that creates a Master Cross-Reference of your entire system.

- **Multi-Lingual**
  Handles C, assembler, or both.
- **Flexible**
  File names with line numbers, or file names alone. Global and local symbols, or globals alone.
- **Powerful**
  Easily handles systems containing 100 source files or more.

VersiMAKE™	$125
VersiCREF™	$75
Both	$150

(Outside U.S., add $5 for shipping and handling)

800-344-4096
(In NJ, 609-779-0784)
MC/VISA/AMEX accepted

SUMMIT INFORMATION SYSTEMS, INC.
496 N. Kings HWY Suite 130
Cherry Hill, NJ 08034

Number **158** on the Reader Service Card

for the PC-compatible family? Specifically, what, if any, cards exist that contain their own processors for graphics primitives such as vectors and arc while still maintaining downward compatibility with the CGA/EGA/VGA standards? I am looking to buy a 386 or 486 platform, hopefully with such a graphics card, for development of a design package that will include features such as spline dragging. This kind of sophistication obviously requires hardware equal to the task.

If you know of any companies that manufacture a graphics card similar to what I would like, it would be much appreciated if you could let me know or even ask them to contact me directly.

David R. Hoy
Newport, Rhode Island

MICHAEL ABRASH RESPONDS: Mr. Hoy almost certainly doesn't need as much power as he thinks. A 34010-based board would probably do the trick, and there are a slew of those, several with VGAs built in or emulated. (I think HP has one, and there are others. Mr. Hoy should look in Computer Graphics World *or in a* PC Magazine *roundup of high-end cards.) An 8514/A might have the power he needs, too, although it doesn't have circle/arc primitives. There are more powerful boards, too—Matrox comes to mind.*

In short, I have no recommendation, because I have no hands-on experience, but there are lots of boards to choose from. In this case, I'd start with the 34010 boards, which are flexible and quite powerful without being ridiculously expensive, and continue the search only if none of those meets the need (but I think one will).

Hope this helps; sorry I don't have a definitive answer. There are 34020 boards just being introduced that are expensive but which should certainly be powerful enough.

On String Searching

The algorithm discovered inside QuickBASIC by Ethan Winer ["QuickBASIC's Fast String-Searching Algorithm," *PJ* 7.6, (November/December 1989)] appears to be essentially the one found on page 198 of *The Bluebook of Assembly Routines for the IBM PC and XT*, by Christopher L. Morgan (New American Library, 1984). It is excellent.

Let me offer a comment about string-searching routines in general: Some are faster than others. However, in general, the subjective impression of speed will depend on how fast you get the output to the screen, not on how fast you find the string. At least this is true for files the size of those normally seen in a microcomputer environment.

For example, when I set out to teach myself about string searching, I wrote a literal translation (from Pascal to assembly language) of the "brute force" algorithm given by Sedgewick on page 243 of his book *Algorithms* (Addison-Wesley, 1983). It used **cmpsb** to compare the first byte of the source string to every byte of the destination in turn, simply because that's what Sedgewick's algorithm said to do. This involved decrementing SI on every nonmatch. I don't suppose you can think of a clunkier, more awkward routine than that if you try for a long time. But with direct screen writing on the IBM PC, running at about 7.8 MHz, it appeared blindingly fast.

Finally, I would like to add my voice to those who want more articles on assembly language. I'm not sure how many will agree with me, but I can think of two areas that could stand some coverage—number conversions, and all sorts of ways of using integer arithmetic wherever possible in preference to floating point. I'm sure most compilers do a good job with calculations, but wouldn't it be true that the requirements of generality in a high-level language would prevent complete optimization for all cases?

You are putting out a splendid magazine. Long may it wave!

David W. Nord
Caledonia, Minnesota

We at *Programmer's Journal* like to hear from our readers. Please address correspondence to Managing Editor, *Programmer's Journal*, 150 N. 4th St., Springfield, OR 97477. You can also contact us via fax (503-746-0071), CompuServe (user ID 76676,427), or MCI Mail (user ID 336-4248). All letters become the property of *Programmer's Journal* and may be edited for publication.

AUTO-HYPHENATION

HY-PHEN-EX is a hyphenation expert system for American English packaged as a C language function. HY-PHEN-EX has over 4800 rules to identify the places where a word may be divided between lines. HY-PHEN-EX comes with full C language source.

HY-PHEN-EX costs only $89.
To order, or for more information, call 415-680-1964 or write to:

GeoMaker Software
P.O. Box 273124
Concord, CA 94527-3124

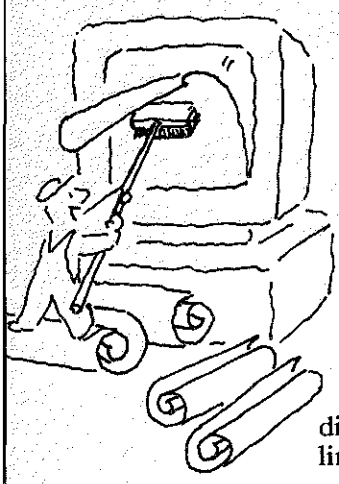
Number **355** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Two great tools.

NEW VERSION 2.8
CLOCK & EDIT UNITS, DIALOG BOXES, AND PROGRESS BARS!

SAYWHAT?!
The lightning-fast screen generator.



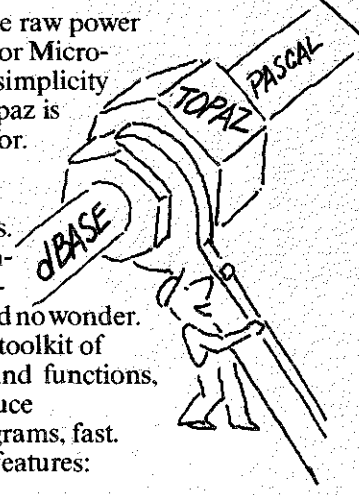
Whether you're a novice programmer longing for simplicity, or a seasoned pro searching for higher productivity, you owe it to yourself to check out Saywhat. You see, with Saywhat, you can build beautiful, elaborate, color-coded screens in minutes! That's right. Truly *fantastic* screens for menus, data entry, data display, and help-panels that can be displayed with as little as one line of code in *any* language.

Here's what you get:

- Design screens, windows, and *moving bar menus!*
- Easy-to-use, powerful editor lets you create screens in a jiffy.
- Pop up your screens and menus with one line of code in dBASE, all the dBASE compilers, your favorite BASIC, Pascal, *or any other language!*
- Screen Library Manager.
- Generates runtime code.
- No runtime license or royalty fees.
- Comes with a 100 page manual, plus dozens of sample programs and free utilities.

$49.95

TOPAZ
The breakthrough DBMS toolkit for Pascal



If you'd like to combine the raw power and speed of Turbo Pascal or Microsoft's QuickPascal with the simplicity and elegance of dBASE, Topaz is just what you're looking for. That's because Topaz was specially created to let you enjoy the best of *both* worlds. The result? You create complete, truly dazzling applications in a very short time. And no wonder. Topaz is a comprehensive toolkit of dBASE-like commands and functions, designed to help you produce outstanding, polished programs, fast. Check out these powerful features:

- Over 220 routines all with easy-to-use, dBASE-like syntax.
- Data entry routines like **SAY, GET, PICTURE, RANGE,** color selection, unlimited data validation.
- Open up to 10 DBF files, with up to 15 indexes with **USE, SELECT, SKIP, APPEND, PACK, INDEX ON, SET INDEX TO,** and **FIND.**
- No need to buy dBASE. **CREATE, BROWSE** and **REPORT** utilities included.
- Easily implement Saywhat and Lotus-style moving bar menus.
- **BROWSE or EDIT** any DBF file with just one line of code! Programmable and windowed too.
- Pick from windowed data or filenames with one line of code.
- Comprehensive Time & Date math in 7 international formats.
- Powerful code and report generators included!
- Comes with a complete 320 page manual, plus sample programs to get you started.

$74.95

ORDER NOW.
YOU RISK NOTHING.

Thousands of satisfied customers have already ordered from us. Why not visit your dealer or call toll-free, right now and put Saywhat and Topaz to the test yourself? They're fully guaranteed. You don't risk a penny.

Special limited-time offer! Save $26. Buy Saywhat?! and Topaz together for just $99 (plus $5 shipping and handling, $10 Canada, $25 International, Calif. residents add 7%).

Visit your nearest dealer
or call toll-free:
800-468-9273
In California: 800-231-7849
International: 415-571-5019

Software Science, Inc.
100 Valley Drive, Brisbane, CA 94005

MONEY BACK GUARANTEE.
If you aren't completely delighted with Saywhat or Topaz, for any reason, return them within 30 days for a prompt, friendly refund.

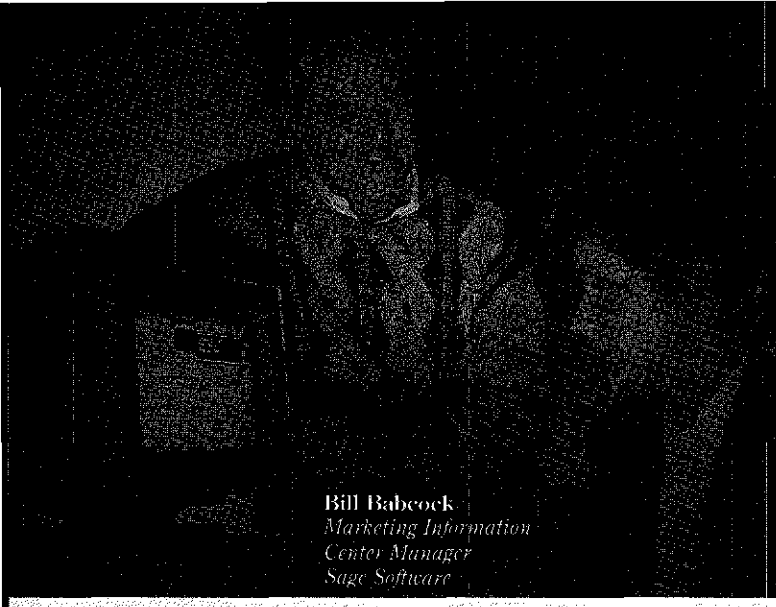
Dealers: SAYWHAT?! and TOPAZ are available from Kenfil Distribution, and in Europe from ComFood Software, W. Germany 49-2534-7093

Guaranteed!

SOFTWARE SCIENCE INC.

Number 198 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Bill Babcock
Marketing Information Center Manager
Sage Software

Q. How can you make sure users love your program before you write it?

A. Demo II, Sage and you.

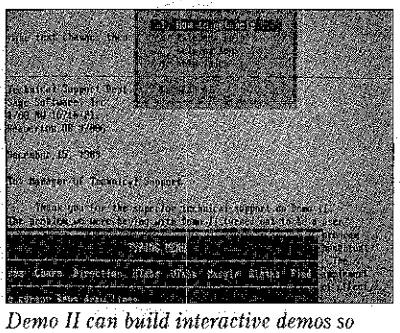
"True story. I'm developing a marketing info system at Sage. Big Project. I've been working with the users for months. I've got notebooks full of data structures, screen drawings, report requirements, and menu designs.

"We're two weeks into coding when my boss gives me a copy of Dan Bricklin's® Demo II.™ 'We're going to acquire this,' he says. So I play with it over the weekend. I prototype my project. On Monday I proudly demo my prototype to the users who *completely* agreed with my paper plans.

"They hate it. Vital information is not accessible from screens where it's needed; frequently used functions are hidden deep in menus; the entry flow doesn't fit the work flow. I don't know whether they did a bad job of telling me what they wanted or I did a bad job of listening.

Polytron has been acquired by Sage Software, Inc. The same team that supports and develops the Polytron products you are familiar with, now supports Demo II.

"But with a few hours of work I fix the problems in the prototype. They try it again. Big smiles. The project is back on track. They love it.



Demo II can build interactive demos so realistic that users can't tell them from finished software. Above is an example of a CUA interface design that will be running on both mainframe terminals and PS/2s.

"The same hands-on demo with real code would have happened months into the project. We would have trashed a lot of work or finished with a system nobody wanted. Demo II saved my project and my time. Now I know why it's so popular."

Dan Bricklin's Demo II can do more than just save you time and trouble. Use it to provide astonishingly realistic demos of applications - self running, interactive, or both. Use it to show management and users that your project is on track. Use it whenever you need to communicate how a program will look, feel, and respond without providing the real code. And though Demo II runs on MS-DOS, many programmers use it to create prototypes for mainframe and minicomputer applications.

Even non-programmers find it easy to learn, but Demo II has so much capability and stores demos so compactly that it's widely used for computer-aided instruction, replacing CAI authoring systems that cost thousands of dollars.

Don't wait until your carefully researched and brilliantly coded project turns out to do the wrong thing the wrong way. The time, the grief, and the reputation you'll save are your own.

$199 U.S. List Price with unlimited run-time distribution rights.

1-800-547-4000

30 Day Money Back Guarantee

SAGE SOFTWARE

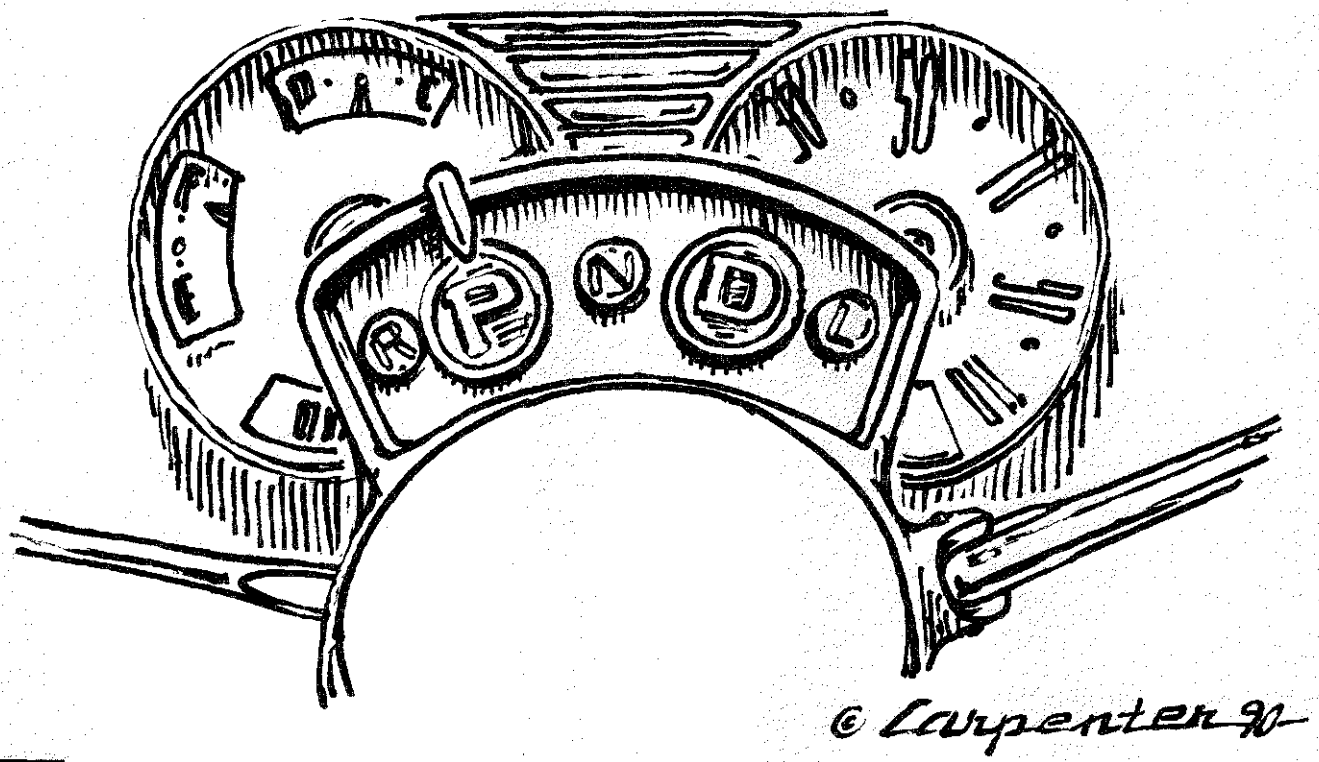
Dan Bricklin's is a reg. trademark of Daniel Bricklin. Requires - 512K IBM PC compatible, Mono., CGA, EGA, VGA or Hercules. MS-DOS 2.0 or later.
Sage Software, Inc., 1700 NW 167th Place, Beaverton, OR 97006, (503) 645-1150, Fax (503) 645-4576, Int'l. (301) 230-3307.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Importance of Good Defaults

by

Brett Glass



Ever since Man the toolmaker first began to build labor-saving devices, he's added defaults to make them easier to use. The word *default* has many meanings, most of them relating to legal and financial situations. In this context, however, a default might best be defined as "that which a tool or system does in the absence of action on the part of the user."

Copyright © 1989 by Brett Glass

Usually, when we talk about using computers and other tools, we don't say much about defaults. We speak about what we *do* do—tasks we explicitly and willfully perform to get a job done. Still, as tools get more complex, we become more and more dependent on good defaults. Defaults become especially helpful in situations where human beings do not have enough arms, legs, and fingers—or fast-enough reflexes—to control everything at once. This situation arises in one of the "interfaces" most of us use every day: the automobile.

Like any complex interface, a car has many defaults that differ from mode to mode. When an automatic-transmission car is in PARK, it stands still if the driver does nothing; in DRIVE, however, the car creeps forward at a slow pace. This is arguably *not* a good default. When I first learned to drive, I was aghast at the idea that simply removing my foot

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

from the brake pedal at the wrong moment could cause serious damage to life, limb, and property! Yet most of us for whom driving is second nature take this default for granted, and few would consider changing it.

What constitutes a good default? There are probably as many ideas on this subject as there are users, but most people will agree on certain general principles. First, the default should reflect the action the user will want to take the majority of the time. (After all, a default isn't much good if one constantly has to override it.)

Second, a system should not default to an action that is potentially highly destructive. Imagine, for instance, that the default argument of UNIX's **rm** (remove file) command was "*" (everything). Not knowing exactly what you're doing on a UNIX box can be dangerous as it is, but this default would certainly discourage new or careless users.

Finally, a default should be easy to override. I once had a bad experience with a hard disk backup program whose default was to restore every file you'd backed up; I lost a substantial amount of work because I wasn't able to tell the program to leave newer versions of files alone.

In the computer world, good defaults can shorten training sessions and create vast productivity improvements. It's especially nice when a single keystroke can trigger the command you'll most likely want to use at any point. Pressing the spacebar in "rn," a UNIX news-reading program, automatically takes you to the next message you haven't read—either by subject or in chronological order. In BIX's CoSy conferencing system, the carriage return key causes a similar default action: reading forward, by reference, or backward, depending on the current reading mode. These defaults work well—they make systems less intimidating for beginners and save users a lot of time and effort compared with systems that require specific commands at many points. In fact, I've designed my own conferencing-system software with the same philosophy.

In the examples above, as in most modal systems, the defaults establish the key differences between the modes. In the Macintosh user interface, the appearance of a modal dialog box changes the results of the usual pointing and clicking actions. In a GEM, Windows, or Presentation Manager dialog box, hitting the ENTER key almost always produces the default action *for that dialog* rather than the action it would perform in the current application.

Many of us are familiar with the built-in defaults that make modern programming languages easier to use. In most versions of BASIC, a variable with a purely alphanumeric name is considered to be a real number unless it is explicitly declared otherwise. Likewise, a variable with a name ending in "$" is automatically a character string, and arrays have a default dimension of 10 if you don't specify how big they are. In FORTRAN, variables whose names began with certain letters,

Now There's a Periscope Board for Your IBM PS/2®

With the new Periscope® Model I/MC, you now have the same robust Periscope Model I debugging capabilities using a PS/2 with Micro Channel® architecture that you already have using a PC, XT, AT, or AT-compatible 80386 machine.

Just like the current Periscope Model I, Periscope Model I/MC has a 32K footprint in system memory, above 640K but in the first megabyte. The board stores the Periscope software and all debugging information (symbols, etc.) in its write-protected RAM.

Designed for use in machines with the IBM Micro Channel bus architecture, the board allows you to add chips to extend the 512K of write-protected RAM to a full two megabytes, if need be. (Most developers find 512K to be quite enough, however.)

Real-time hardware-assisted debugging of programs running on PS/2s is now possible! The remote feature of the new Version 4.3 Periscope software enables Periscope IV to support real-time debugging of programs running on DOS-based machines, including those with Micro Channel architecture. The open architecture remote debugging feature will support OS/2® and other protected-mode environments in the near future. Call for details.

Don't worry about trashing your debugger, debugging large programs, or erratic bugs.

With this new board in your IBM PS/2 or compatible, Periscope uses zero memory in the lower 640K. So you don't have to worry about things like a runaway program trashing your debugger, or not being able to debug a very large program, or having bugs appear or disappear when you load your debugger.

Use the break-out switch, which plugs into the board, to break in to your system safely any time. It keeps you from having to power down and back up when your system hangs. You can just press the little red "panic" button to find out exactly what is going on.

Periscope Model I for PCs, XTs, ATs, and AT-compatible 80386s. The manual, disk, and quick-reference card shown come with all models of Periscope.

Choose from a full line of professional software and hardware-assisted models.

All models include Version 4.3 software, manual, and:

- *New* **Periscope I/MC** (MC Board for short) has 512K Micro Channel-compatible board & break-out switch ..... $745.
- **Periscope I** has 512K PC- and AT-compatible board & break-out switch ..... $595.
- **Periscope II** has break-out switch .... $175.
- **Periscope II-X** has no hardware ..... $145.
- **Periscope III** has PC- and AT-compatible real-time board (to 10MHz) & break-out switch ..... $1395.
- **Periscope IV** has 80286 and 80386 AT-compatible real-time hardware (to 25MHz) & breakout switch ..... $2195-$2995.
- **PLUS** board is Model I board (no software), optional with Models III & IV ..... $500.

Call Toll-Free Today For More Information 800-722-7006

MAJOR CREDIT CARDS AND QUALIFIED COMPANY PURCHASE ORDERS ACCEPTED

IBM, PS/2, OS/2, and Micro Channel are registered trademarks of the IBM Corporation.

PC MAGAZINE MACRO AWARDS FINALIST

1197 PEACHTREE ST. • PLAZA LEVEL • ATLANTA, GA 30361
404/875-8080 FAX 404-872-1973

The Periscope Company, Inc.

Number 6 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

such as *I* and *J*, are implicitly integers; others are implicitly reals. If you don't like these defaults, you can change them with the IMPLICIT statement. In the C language, each function is expected to return an integer unless it is explicitly declared as returning another type.

Languages, of course, are the subject of endless controversy, and opinions vary regarding the usefulness of these defaults. Some may claim that implicit typing leads to sloppy programming and makes programs less readable, but for short programs—or for programming situations where you don't want to break your train of thought, move to a different part of the file, and add a declaration—such defaults can be invaluable timesavers. And for those who want complete lists of declarations, the compiler can certainly be designed to add them. (The compiler might even be set up to demand that the programmer document each one—a feature that would likely be turned off by default!)

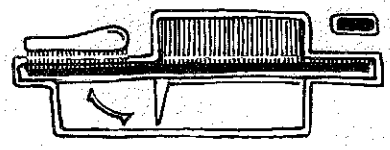
Another group of programming languages, those with object-oriented features, is at last coming into vogue after years on the fringes of the software community. A key feature of object-oriented languages called *inheritance* allows the subprograms that work with one class of objects to work also on objects derived from it—unless this feature is explicitly overridden. In other words, object-oriented programming environments owe much of their power, flexibility, and convenience to powerful sets of defaults! Behaviors acquired via inheritance can save a programmer days or even weeks of tedious work reimplementing algorithms for a new object type.

Defaults work best when they become second nature to us. Many of the current conflicts about ownership of graphical user interfaces (if indeed they can be owned) revolve around such "learned" defaults. It's easy to deal with an interface when you've come to understand it on a gut level; it's tragic that microcomputer makers, like minicomputer manufacturers in their heyday, now seek to discourage users from switching by demanding that the others present different interfaces with different defaults.

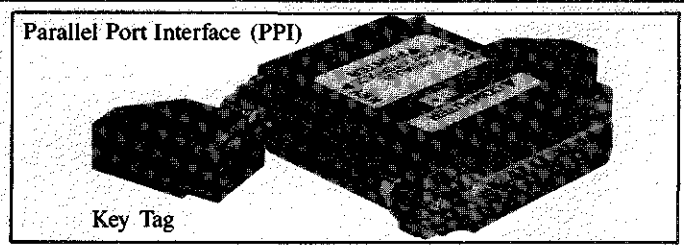
These companies contend so mightily, and at such great expense, because they recognize that defaults lead us to develop complementary habits—and habits, once developed, are hard to change. Will a user who is used to a menu at the top of the screen be thrown off balance by one that pops up instead? Will someone who is accustomed to discarding files in a "trash can" recognize that the "black hole" icon on another system performs the same function? Imagine the chaos that would ensue if every brand of automobile had an entirely different user interface, and you'll understand why I fervently hope that the courts will allow computer manufacturers to freely adopt the same defaults.

Defaults are easy to forget—good ones especially so. But even bad defaults eventually become habit, and it's a rewarding—if challenging—task to reexamine the defaults in tools you use daily. What other common devices have defaults that could be improved? I'll leave you to ponder this question while I hasten to rescue a batch of bran muffins from destruction. They will surely burn if I don't take them out of the oven, since the oven is now in a mode where it will continue to cook—by default—unless I shut it off. PJ

Brett Glass is a free-lance writer, programmer, and consultant residing in Palo Alto, Cal. He can be reached at rogue@well.sf.ca.us on Internet, on CompuServe at 72267,3673, or as GLASS on BIX.



World Class Software Security



The parallel port interface (PPI) connects between the printer port on a PC and the printer cable. The PPI holds two Key Tags, one on each side. Each Key Tag contains a secure custom chip which is pre-programmed by Glenco to only work with the assigned software package. A second Key Tag can be employed to protect another package, or may be used to turn other software packages "on", remotely or on-site.

- STANDARD KEY TAG - Software is protected for an unlimited number of executions. They are pre-programmed to include a sequentially assigned S/N.
- COUPON KEY TAG - Software is valid for a preset number of executions. The Coupon count can be reset remotely or on the customers site by using a second update Key Tag.
- READ/WRITE KEY TAG - With programmable memory. Perfect for companies which have multiple products or a product with several optional modules. By having several packages protected using one Key Tag, your costs are lowered.
- DURATION KEY TAG - Has a clock on board. (Available late '89)

Secure software and data with reliable, effective protection products that won't burden honest users.

Glenco is a world leader in the area of software security products and services. Our copy protection products and data security products are second to none. They are designed to function on a wide variety of third party hardware. We have over 3500 satisfied software firms utilizing our products. We also have a full line of disk based protection systems.

- MACHINES SUPPORTED - IBM PC/XT/AT & PS/2, Macintosh
- OPERATING SYSTEMS - MS-DOS, XENIX, Network, Finder, & Multifinder.
- LANGUAGES/COMPILER - Over 50, including runtime packages, data bases and spread sheets. We have a non-programmers interface as well.

Call or write for more information.

GLENCO ENGINEERING INC.
SERVING THE SOFTWARE INDUSTRY SINCE 1979

721 W. Algonquin Road, Arlington Hts., IL 60005, (312) 364-7638, FAX 364-7698

In Europe contact: SDC Security Systems, The Netherlands
Tel: +31-45-441535, FAX: +31-45-444747

Number 323 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

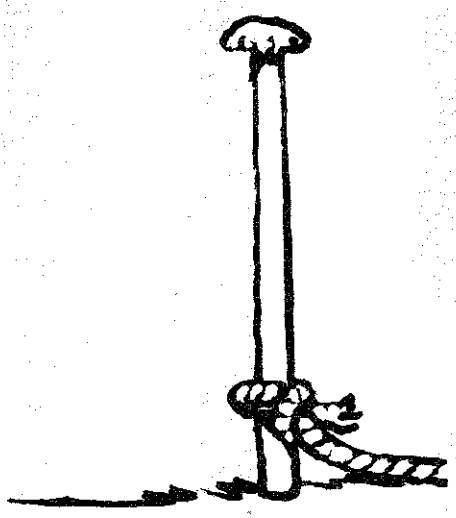
Faster Circles for the VGA

by
Michael Abrash

Two years back, I set out to write the fastest possible line-drawing code for the PC. I took Bresenham's algorithm apart piece by piece until I understood it completely, and then I unleashed all the assembly language skills I had on the task of implementing that algorithm. When I was done, I had boosted performance by about two times over a standard Bresenham's assembly language implementation, and I couldn't for the life of me imagine how the code could be improved one iota. In other words, I was confident that I had written just about the fastest code possible.

That was an incorrect conclusion—to put it mildly. Over the past few months, it seems as if every graphics programmer I know has decided to clue me in on his or her favorite way to draw lines. I've heard about state-machine line drawing, run-based line drawing, fixed-point line drawing, and even a technique that lets the floating-point processor perform a single division while the line-drawing routine sets up and then uses the bits of the floating-point remainder to control line drawing. Wildly different as these approaches are, they have one thing in common—they're all faster than my original "optimized" code.

Which brings us to today's installment. Last time, we learned how to draw circles and then vastly improved performance by switching from floating-point-based code to Hal Hardenbergh's integer-only algorithm, which requires multiplication and division only during the initial setup. That was step 1 of graphics optimization: selection of algorithm. Step 2 (matching the algorithm to the hardware) and step 3 (conversion of critical code to assembly language) are coming up next. When we're done, we'll have circle-drawing code that's more than 20 times as fast as the original implementation. Our final circle-drawing code will be more than 3 times as fast as the faster routine we developed last time, *even though both routines use exactly the same algorithm.* Clearly, there is more to performance than selecting the right algorithm, although that is an essential starting point.



Fast as our final implementation will be, however, I wouldn't dream of calling it "optimized." As the story that began this installment indicates, it's presumptuous indeed to think you've come up with the best possible graphics code, and circle drawing is no exception. In fact, the code could easily be speeded up by using in-line code to draw multiple pixels without looping, and it could possibly be made faster by using fixed-point arithmetic or by encoding the pixel list in a more readily usable format. Many small improvements—simplified calculations, eliminated redundancies, and the like—could be made throughout the code, although optimizing outside the inner loops tends to be a waste of time and effort.

My point is don't take the code in this installment as gospel—it's fast, but it could be at least a little faster, and maybe a lot faster. Understand what I've done, and then try to do it better. One of the great things about PC graphics programming is that it's

Installment 17: In which we draw circles fast and faster—but surely not fastest.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

almost always possible to do exactly that.

Because there's a *lot* of code in this installment, I have less space than usual for text—I'll have to move fast and go light on the explanations. So let's get going.

Faster Circle Drawing

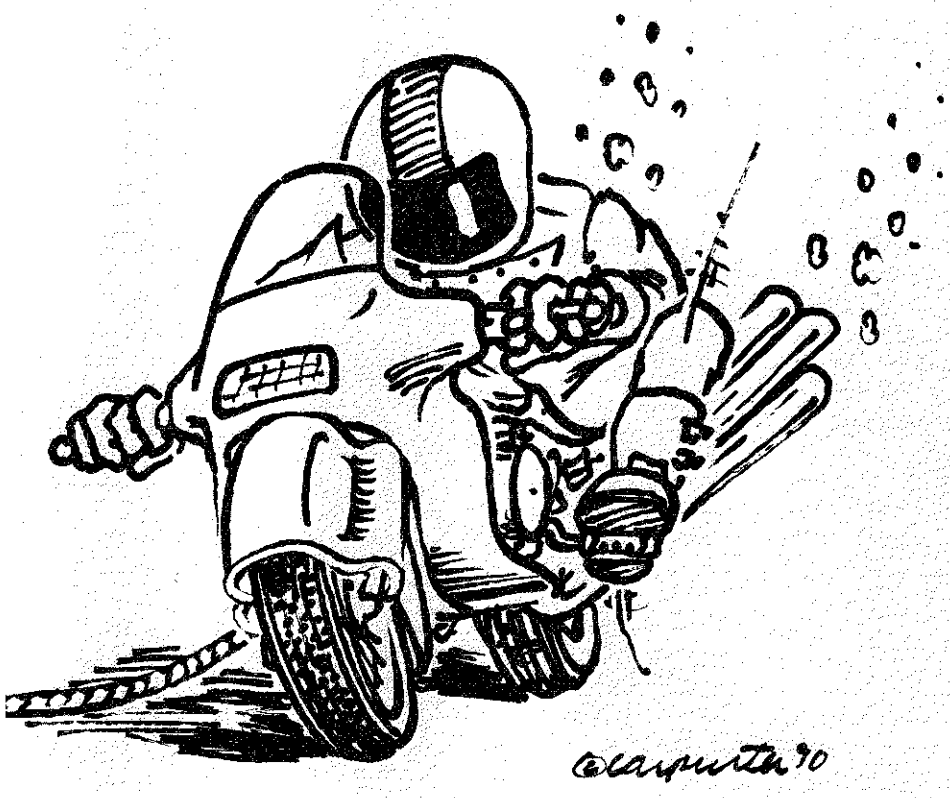
In focusing on specific optimizations in this installment, I'm going to assume that you're familiar with circle drawing in general. If not, I suggest you read the preceding installment, which discussed the fundamentals of circle drawing and described in detail the integer-only algorithm we'll use in this installment.

The question at hand is "How can we speed up the circle-drawing code we saw at the end of the preceding installment, which used integer-only calculations and required no multiplication or division inside the main loop?" If you refer back to that code [*PJ* 8.1, pp. 28–31], you'll see that while the main loop is quite compact, it calls a separate routine to draw each pixel, and it's there that we can save a slew of processor cycles.

The pixel-drawing routine in the last installment was short, but it performed two time-consuming and avoidable tasks. First, it calculated each pixel's display-memory offset from scratch, requiring a multiply and the construction of a far pointer. Similarly, it generated each pixel's bit mask from scratch with a little arithmetic and a potential multibit shift. As it turns out, neither of those actions is actually necessary, because with this circle-drawing approach, the offset and the mask for a given pixel are incremental modifications of the offset and mask for the previous pixel.

In other words, once we've drawn pixel *n*, we can generate the bit mask for pixel *n*+1 with, at most, a single-bit rotation. Likewise, we can calculate the offset for pixel *n*+1 with, at most, an increment or decrement followed by an addition. Those simpler calculations can save many cycles in comparison with calculating each pixel from scratch. We can also save some cycles by moving the code that draws each pixel into the drawing loop and eliminating the call to and return from the drawing function.

There's another, less obvious optimization we can make. Consider this: The process of calculating adjacent points along one-eighth of a circle is nothing more than a matter of deciding whether to move along both axes after each pixel or only along the major axis; making these moves and drawing the corresponding pixels illuminates precisely those pixels closest to the desired arc. Given that, it should be clear that it's possible to calculate the set of moves from one pixel to the next, store that set in an array, and then play back the array with varying major- and minor-axis directions eight times to draw the circle. Put another way, we could calculate and store the information needed to determine when to advance along the minor axis just once for a given



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Listing	Compiler/Processor			
	Microsoft C 5.0		Turbo C 2.0	
	286	386	286	386
Old-1 (C/floating point)	468 sec	175 sec	339 sec	127 sec
Old-3 (C/integer)	44 sec	16 sec	47 sec	18 sec
1 (C/incremental pixel addressing)	31 sec	12 sec	32 sec	13 sec
3/4 (ISVGA=0) (ASM)	14 sec	7 sec	14 sec	7 sec
3/4 (ISVGA=1) (ASM/write mode 3)	13 sec	5.5 sec	13 sec	5.5 sec

Table 1. The execution times of the various circle-drawing implementations in this installment and the last when linked to Listing 2. (Listings titled "Old-*n*" are from the last installment.) Maximum optimization (*/Ox* for Microsoft C, *-G -O -Z -r* for Turbo C) was used to compile all C code. Times in the columns labeled "286" were recorded on a Video Seven VRAM VGA running on a 10-MHz, 1-wait-state AT clone (a monochrome adapter was also installed, making the VGA an 8-bit device); times in the columns labeled "386" were recorded on a built-in Paradise VGA in a 20-MHz, 32K–0-wait-state-cache Toshiba 5200 (a monochrome adapter was also installed). No floating-point processor was installed in either computer. Results could vary considerably on different hardware.

circle and then use that information eight times to draw the eight symmetric arcs that make up the circle, interpreting the information slightly differently each time.

What does that buy us? It lets us separate the arc-calculation code and the arc-drawing code so that each can be better optimized. Truth to tell, that's not such a big deal in C; we could just calculate each point and draw all eight symmetries on the spot, as we did in the earlier installment (although that would require maintaining eight display-memory pointers and eight masks). When we get to assembly language, however, separation of calculation and drawing will stand us in good stead, for by reducing complexity it will allow us to keep all variables in registers for the duration of both the calculation and the drawing loops—and that will translate directly into better performance.

I don't know whether the separation of calculation and drawing has a similarly large advantage in C—or if it has any advantage at all. However, given that the major purpose of the C code in this installment is to prototype and illuminate the assembly language code, I'll use the same approach of separating calculation and drawing in the C code that I'll use later in the assembly language code.

Listing 1 shows a C circle-drawing function that uses the separate calculation-and-drawing approach, along with the incremental-offset–and–mask-calculation technique described earlier. Arcs with horizontal major axes are handled separately from arcs with vertical major axes, because the process of advancing across a scan line of EGA/VGA memory is fundamentally different from that of advancing from one scan line to the next. When compiled and linked to Listing 2, which repeats the drawing of a set of circles 20 times for timing purposes, Listing 1 is about 40 percent faster than Listing 3 in the preceding installment (henceforth called Listing Old-3), which was our speed champ until now, as shown in Table 1. That's not the 7-to-11-times improvement that Listing Old-3 produced over Listing Old-1, but it's certainly significant.

It's worth noting that listings 1 and Old-3 both use the integer-only algorithm presented in the last installment. The performance difference between the two listings comes from understanding the code the compiler generates for each and the performance implications of that code. In particular, Listing 1 virtually eliminates the 8086's relatively slow multiplication, multibit-shift, and call and return instructions.

C Implementation and a Bug Fix

Listing 1, like Listing Old-3, uses long integers because the squared values used can grow too large for short integers. You can avoid the performance cost of long integers by making special cases of circles with radii less than 256. For the same circles to work with short integers, however, short *unsigned* integers must be used; otherwise, comparisons may not work correctly when squared quantities exceed 32,767.

In fact, there is a correction along these lines to be made to Listing Old-3: In the initial calculation of **MinorAxisSquaredThreshold**, **MinorAxis** should be cast to an unsigned long, in order to avoid the squared result being interpreted incorrectly as a signed integer value. Likewise, **Radius** should be cast to an unsigned long in the initial calculation of **RadiusSqMinusMajorAxisSq**. Neither of these omissions affected drawing accuracy with the sample program of Listing Old-2, but circles with radii greater than 255 will not be drawn correctly without this bug fix to Listing Old-3. Fortunately, Listing Old-3 was primarily illustrative; you'll want to use the routines from this installment.

Listing 1 (and, indeed, all the listings in this installment and the last) assumes that no clipping is needed. If that's not the case, the approach of generating a pixel list and then

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

drawing the eight symmetries from the list could lend itself rather well to clipping, because the drawing routine could use the list to skip quickly over any initial portion of an octant that's outside a clip region and could terminate processing immediately when the far edge of the clip region is reached. At the very least, clipping from a pixel list is surely more manageable than attempting to draw and clip all eight symmetries simultaneously, and it is undoubtedly more efficient than calculating and drawing the clipped arcs for each of the eight octants separately.

Putting the Pedal to the Metal

As regular readers of this column know, it's my belief that the critical code for graphics primitives should be written in assembly language. Circle drawing is no exception. Listings 3 and 4 show a hybrid C–assembly language circle-drawing routine that is yet a good deal faster than Listing 1; as Table 1 shows, the code in listings 3 and 4 is about twice as fast as Listing 1's code.

Why didn't I use pure assembly language? For one thing, it's rarely worth using assembly language outside heavily used loops—the overall performance improvement is generally imperceptible and doesn't justify the extra work. For another, the listings are long as it is; I didn't want to try the patience of the editors further with a hundred or so additional lines of code. All in all, listings 3 and 4 strike a good balance between performance, ease of coding and comprehension, and maintainability.

Listings 3 and 4 generally correspond directly to Listing 1, although that may be obscured by the considerable optimization performed in Listing 4. Two aspects of listings 3 and 4 do not correspond to Listing 1, however, and bear further discussion.

When the major axis is horizontal, multiple horizontally adjacent pixels are often drawn. Whenever multiple adjacent pixels are controlled by the same display-memory byte, it is possible to draw all the pixels that reside in that byte with a single display-memory read/write operation. This is highly desirable, because display memory is extremely slow compared with normal system memory, especially on 386-based computers. Consequently, Listing 4 accumulates pixels that reside in the same byte when drawing horizontal major-axis arcs and accesses display memory only when all pixels that could possibly belong in a given byte have been processed. (See "Fast Line Drawing for the EGA and VGA" in *PJ* 7.3 [May/June 1989] for the application of this technique to straight-line drawing.)

Pixel accumulation is a good example of matching an algorithm to the hardware. The basic operation of the algorithm is unchanged, but the code is fine-tuned so that the implementation suffers less from the poor performance of display memory.

Listings 3 and 4 contain another example of matching the circle-drawing algorithm to the hardware: optional support for write mode 3, which only the VGA supports. When the ISVGA symbols in listings 3 and 4 are set to 1 (make sure both symbols are the same at all times), write mode 3 is used to draw pixels, improving overall performance by as much as 25 percent. ISVGA makes Listing 4 in particular a little hard to follow, but I think you'll agree that the performance improvement and the exposure to a useful VGA-specific optimization are worth the trouble.

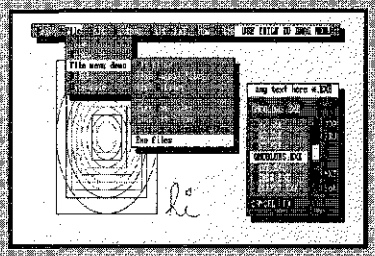
The virtue of write mode 3 is that it **and**s the Bit Mask register with the byte written by the CPU to form the working bit mask; since there's no need to do an **out** when write mode 3 is used, you eliminate both an instruction and the wait states that occur during I/O to most VGAs. What's more, by eliminating the need for **out**, we free up AL and DX for other purposes, in this case making it possible to use **lodsb**. (See "Write Mode 3 of the VGA" in *PJ* 6.1 [January/February 1988] for more information on write mode 3.)

For write mode 3 to work properly, the desired bit mask must be written to memory. (The set/reset circuitry provides the pixel color.) First, however, display memory must be read to latch the surrounding pixels so that the bit mask can

The professional menu system in a *graphics* environment for TURBO C, MICROSOFT C, TURBO PASCAL:

***graphics*-MENU** *with* **Data Entry**

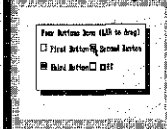
Starts at $99 (BGI) SOURCE AVAIL.



Now Available: **Foreground/ Background Tasks**

graphics-MENU™ from ISLAND SYSTEMS is a comprehensive library plus utilities package that allows the developer of graphics software applications to quickly create a user-friendly interface. Using *graphics*-MENU allows you to spend more time focusing on the internals of your application.

- Several types of menus including horizontal bar & vertical pull-down menus, button menus, list menus (w/ scroll bar), gauge menu, analog clock menu & form menus (with opt. Data Entry Module).
- Message boxes with or without user prompt.
- Additionally included are items such as mouse support functions and geometric functions.
- Features include: EMS support, foreground/ background task capability, automatically saves/restores underlying graphics image, shadows for 3d effect, most items draggable by the mouse, menus can work with mouse or keyboard or both.
- CASE tools (**Menu Designer**™ and **Data Entry Designer**™) provide on-screen WYSIWYG design environments for menu creation and include automatic code generation.
- Utilities **CurEdit**™ (creates custom cursor icons) and **Color Customizer**™ (on-screen palette for custom menu colorizing) are included in the basic *graphics*-MENU package.



*MenuDesigner included with META version. Available for $49 with BGI version. Data Entry Designer is an optional add-on for $49.

Data Entry

The **Data Entry Designer**™ enables creation of forms menus via an on-screen design environment and then writes TP or TC/MS-C code to implement the form in your application.

The **Data Entry Module**™ is a specialized library for creation of forms menus. A high degree of flexibility has been built-in to allow "hooks" into the library for all types of data entry applications. *Features:* dBASE-like string specification, real-time data checking, data type validation, automatic field movement, several types of form exit conditions, help message facility, professional string editing.

** 30 day money back guarantee **

BGI VERSION
Requires: Borland Turbo Pascal or Turbo C, IBM with graphics, hard disk.
graphics*-MENU *includes: graphics-MENU library, CurEdit, Color Customizer. **Specify** Turbo Pascal or Turbo C when ordering.

BGI *graphics*-MENU .....$ 99
MenuDesigner.....$ 49
Data Entry Designer with Data Entry Module.....$ 99
library source.....$ CALL

META VERSION
Requires: MetaWindow (from Metagraphics Software), IBM with graphics, hard disk.
graphics*-MENU *includes: graphics-MENU library, CurEdit, Color Customizer &MenuDesigner. **Specify** TP, TC or MS-C

META *graphics*-MENU.....$149
Data Entry Designer with Data Entry Module.....$ 99
library source .....$ CALL

Island Systems
7 Mountain Rd, Burlington MA 01803

CALL: (617) 273-0421
FAX: (617) 933-1152
DEMO AVAILABLE

Number **317** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ Code Is Available!

On disk from:

Programmer's Journal
P.O. Box 70167
Eugene, OR 97401-0110
Call toll-free:
800-234-0386
In Oregon call:
503-747-0800
All Code Disks $10.00 each.
To order, see ad on page 83.

PJ Code DiskSeriptions are available for only $39.95.
To order, see ad on page 8.

On BBS from:

Buffer Board
538 Cardinal Way
Springhill, OR 97477
(Voice) 503-741-1414
(Data) 503-747-7636

EDI
1338 Hermosa Ave. #7
Hermosa Beach, CA 90254
213-374-3479
Call for your local access number.

Generation Five Inc.
1295 Tech Road
Silver Spring, MD 20904-9988
(Voice) 301-680-9300
(Data) 301-680-9300
Accessible via PC Pursuit.
Washington, D.C.
From Files submenu, select
"P" for Personal Downloads.

Gilmore Systems of Southern California
5442 Indian Hills Drive
Simi Valley, CA 93063
(Voice) 805-582-1380
(Data) 805-582-9306

Ithaca Street BBS
1145 Ithaca
Boulder, CO 80303
(Voice) 303-494-8865
(Data) 303-494-8866

Logical Alternatives Incorporated
Calder Square
P.O. Box 10674
State College, PA 16805-0674
(Voice) 814-234-9088
(Data) 814-234-1881

Mnematics Videotex
P.O. Box 19
Spring, NY 10976-0019
(Voice) 800-322-3693
(Voice) 914-355-0194 in N.Y.
Mention PJ when you call.

PCS-49 on CompuServe
Section 10
Sponsored by:
MISOSYS, Inc.
P.O. Box 239
Sterling, VA 22170
(Voice) 703-450-4181

The Silicon Zone
820 Tivoli Circle #203
Deerfield Beach, FL 33411
(Voice) 305-360-9362
(Data) 305-360-9376

work its magic. The most efficient way to accomplish both reading from and writing to display memory is to issue a single instruction with **and**, **or**, or **xchg**. However, **xchg** would wipe out the register containing the working copy of the bit mask, and **and** with any value other than 0FFh or **or** with any value other than 0 would alter the value written to memory so that it would no longer set the desired bit mask. This is solved by selecting read mode 1 (color-compare mode) and setting the Color Don't Care register to 0, with the result that 0FFh is always read from display memory. Once that is done, we can **and** the desired bit mask with display memory without altering either the value written to memory or the register containing the working copy of the bit mask. (See "Reading VGA Memory" in *PJ* 6.4 [July/August 1988] for a discussion of this application of read mode 1.)

By the way, listings 3 and 4 run just as well on VGAs as on EGAs if both ISVGA symbols are set to 0—they just run more slowly than if compiled/assembled specifically for the VGA.

There's no reason that the C code in Listing 1 couldn't be made faster by using both the write mode 3–read mode 1 and pixel-accumulation techniques we've applied in the assembly language code. However, such approaches, which save an approximately fixed number of cycles, have a greater-percentage impact on overall performance after all other aspects of the code have been streamlined, and so they are most worth using in high-performance assembly language code.

When graphics code is truly streamlined across the board, as in Listing 4, the characteristics of the display adapter can affect performance significantly. For example, the 286 timings in Table 1 were all performed on a 10-MHz AT with both a Video Seven VRAM VGA and a monochrome adapter installed. It's a little-known fact that when a monochrome adapter is present, all VGAs must revert to being 8-bit memory devices. When the monochrome adapter was removed from the test-unit AT, allowing the VRAM VGA to become a 16-bit device, the time for the non-VGA-specific version of listings 3 and 4 dropped from 14 seconds to 12.5 seconds—an improvement of more than 10 percent. Listing 4 has reached the point where it is starting to bump up against the hardware's inherent limits—a sure sign of high-performance code and an indication that perceptible performance variations from one make of VGA to the next are likely to occur.

In Conclusion

In the last two installments, we've seen performance improvements of as much as 36 times from our initial circle-drawing implementation. Viewing this in terms of the three optimization steps I described at the outset, that breaks down to, very roughly, 5 to 10 times improvement from algorithm selection, 1.5 to 2.5 times improvement from matching the algorithm to the hardware, and 1.5 to 2 times improvement from conversion to assembly language.

The conclusions are two:

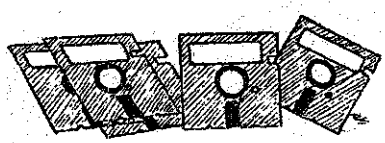
- Algorithm selection is indeed important, but processor- and hardware-specific optimizations are important, too.
- Circles, the slowest of graphics primitives, can be drawn surprisingly rapidly—in the ballpark with, if not quite so fast as, lines.

Feedback and suggestions are welcome, as always. Next time, it's on to ellipses.

See you then. PJ

Contributing Editor Michael Abrash is a senior engineer working on high-performance graphics software for Metagraphics, in Scotts Valley, California. Michael's book Power Graphics Programming, *an anthology of his "On Graphics" articles from* Programmer's Journal, *published by Que Corporation and* PJ, *is available in bookstores and by mail order (see page 30). Michael is also the author of* Zen of Assembly Language, *published by Scott, Foresman & Company. Readers may write to Michael at P.O. Box 414, Burlington, VT 05402-0414, or contact him via MCI Mail, user name* MABRASH.

Code follows.



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

More power, more performance, more control with the. . .

ADVANCED C PROGRAMMER'S LIBRARY

. . . only $4.95

— a $94.85 value —

as your introduction to the

Library of Computer and Information Sciences

You simply agree to buy three more books — at significant savings — within the next 12 months

C PROGRAMMER'S TOOLKIT

Jack Purdum

— includes a 5-1/4" disk —

Make your programs more versatile, more powerful, and easier to create with this invaluable toolkit. Packed with time-saving screen and memory functions as well as statistical and financial routines. You get 320 pages of tips, tricks, and ideas along with a 5-1/4" utility-packed disk.

Fully compatible with ANSI C, this step-by-step manual covers essential functions and routines including: functions drawn from MS-DOS resources (for example, interrupt calls to the operating system), functions based on hardware elements such as direct writing to screen memory, and generic functions for any ANSI C compiler such as statistical and financial routines.

Publisher's price: $39.95

C WITH ASSEMBLY LANGUAGE

Steven Holzner

Strengthen your C programs with assembly language. This example-filled guide shows you how to write powerful C programs that run as quickly as the C-assembly language interface, and actually incorporate assembly language instructions into your routines.

You learn how to write stand-alone assembly language modules, how to use inline assembly language and pass parameters, how to link external files and assembly language functions to C, and much more.

Publisher's price: $29.95

FRACTAL PROGRAMMING IN C

Roger Stevens

This extraordinary guide helps you reproduce — and create your own — complex fractals on a personal computer. You get an invaluable library of C routines for creating a myriad of fractals and images, as well as compelling ways to apply fractals to physical sciences.

Along with a user-friendly review of fractals, you'll get dazzling illustrations and easy-to-use routines for an extraordinary collection of useful and fascinating fractals. You'll have images ranging from Lonenzo and other strange attractors to population and bifurcation diagrams, from the snowflake and other von Koch curves to Peono curves, and much more.

Publisher's price: $24.95

MEMBERSHIP BENEFITS• In addition to getting the Advanced C Set for only $4.95 when you join, you keep saving substantially on the books you buy. • Also, you will immediately become eligible to participate in our Bonus Book Plan, with savings of 60% off the publishers' prices. • At 3-4 week intervals (15 times per year), you will receive the Library of Computer and Information Sciences News, describing the coming Main Selection and Alternate Selections, together with a dated reply card. • In addition, up to two times a year, you may receive offers of Special Selections which will be made available to a group of select members. • If you want the Main Selection, do nothing, and it will be sent to you automatically. • If you prefer another selection, or no book at all, simply indicate your choice on the card and return it by the date specified. • You will have at least 10 days to decide. If, because of late mail delivery of the News, you should receive a book you do not want, we guarantee return postage.

The Library of Computer and Information Sciences is the oldest and largest book club especially designed for computer professionals. In the incredibly fast-moving world of data processing, where up-to-the-moment knowledge is essential, we make it easy to keep totally informed on all areas of the information sciences. What's more, our selections offer you discounts of up to 30% or more off publishers' prices.

The Library of Computer and Information Sciences
Delran, New Jersey 08075

Please accept my application for trial membership and send me the Advanced C Set (00046)billing me only $4.95, plus shipping and handling. I agree to purchase at least three additional Selections or Alternates over the next 12 months. Savings range up to 30% and occasionally even more. My membership is cancelable any time after I buy these three additional books. A shipping and handling charge is added to all shipments.

No-Risk Guarantee: If I am not satisfied—for any reason— I may return the Advanced C Set within 10 days. My membership will be canceled, and I will owe nothing.

Name _____

Address _____ Apt. ___

City _____

State _____ Zip _____

(Books purchased for professional purposes may be a tax-deductible expense. Offer good in Continental U.S. and Canada only. Prices slightly higher in Canada.)

Programmer's Journal 3-4/90 7-FZ2

Number **342** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

All code in this issue is available through our bulletin board services or directly from *PJ* on disk. See pages 8, 88, 22, and 93.

Code from "Faster Circles for the VGA," by Michael Abrash

1

LISTING 1

```
/* Draws a circle of the specified radius and color, using a fast
 * integer-only & square-root-free approach, and generating the
 * arc for one octant into a buffer, then drawing all 8 symmetries
 * from that buffer.
 * Compiles with either Turbo C 2.0 or MSC 5.0.
 * Will work on VGA or EGA, but will draw what appears to be an
 * ellipse in non-square-pixel modes.
 */
#include <dos.h>

/* Handle differences between Turbo C & MSC. Note that Turbo C accepts
   outp as a synonym for outportb, but not outpw for outport */
#ifdef __TURBOC__
#define outpw  outport
#endif

#define SCREEN_WIDTH_IN_BYTES 80    /* # of bytes across one scan
                                       line in mode 12h */
#define SCREEN_SEGMENT        0xA000 /* mode 12h display memory seg */
#define GC_INDEX              0x3CE /* Graphics Controller port */
#define SET_RESET_INDEX       0     /* Set/Reset reg index in GC */
#define SET_RESET_ENABLE_INDEX 1    /* Set/Reset Enable reg index
                                       in GC */
#define BIT_MASK_INDEX        8     /* Bit Mask reg index in GC */

unsigned char PixList[SCREEN_WIDTH_IN_BYTES*8/2];
                                    /* maximum major axis length is
                                       1/2 screen width, because we're
                                       assuming no clipping is needed */

/* Draws the arc for an octant in which Y is the major axis. (X,Y) is the
   starting point of the arc. HorizontalMoveDirection selects whether the
   arc advances to the left or right horizontally (0=left, 1=right).
   RowOffset contains the offset in bytes from one scan line to the next,
   controlling whether the arc is drawn up or down. Length is the
   vertical length in pixels of the arc, and DrawList is a list
   containing 0 for each point if the next point is vertically aligned,
   and 1 if the next point is 1 pixel diagonally to the left or right */
void DrawVOctant(int X, int Y, int Length, int RowOffset,
   int HorizontalMoveDirection, unsigned char *DrawList)
{
   unsigned char far *ScreenPtr, BitMask;

   /* Point to the byte the initial pixel is in */
#ifdef __TURBOC__
   ScreenPtr = MK_FP(SCREEN_SEGMENT,
      (Y * SCREEN_WIDTH_IN_BYTES) + (X / 8));
#else
   FP_SEG(ScreenPtr) = SCREEN_SEGMENT;
   FP_OFF(ScreenPtr) =(Y * SCREEN_WIDTH_IN_BYTES) + (X / 8);
#endif
   /* Set the initial bit mask */
   BitMask = 0x80 >> (X & 0x07);

   /* Draw all points in DrawList */
   while ( Length-- ) {
```

2

```
      /* Set the bit mask for the pixel */
      outp(GC_INDEX + 1, BitMask);
      /* Draw the pixel. ORed to force read/write to load latches.
         Data written doesn't matter, because set/reset is enabled
         for all planes. Note: don't OR with 0; MSC optimizes that
         statement to no operation */
      *ScreenPtr |= 0xFF;
      /* Now advance to the next pixel based on DrawList */
      if ( *DrawList++ ) {
         /* Advance horizontally to produce a diagonal move. Rotate
            the bit mask, advancing one byte horizontally if the bit
            mask wraps */
         if ( HorizontalMoveDirection == 1 ) {
            /* Move right */
            if ( (BitMask >>= 1) == 0 ) {
               BitMask = 0x80;    /* wrap the mask */
               ScreenPtr++;       /* advance 1 byte to the right */
            }
         } else {
            /* Move left */
            if ( (BitMask <<= 1) == 0 ) {
               BitMask = 0x01;    /* wrap the mask */
               ScreenPtr--;       /* advance 1 byte to the left */
            }
         }
      }
      ScreenPtr += RowOffset; /* advance to the next scan line */
   }
}

/* Draws the arc for an octant in which X is the major axis. (X,Y) is the
   starting point of the arc. HorizontalMoveDirection selects whether the
   arc advances to the left or right horizontally (0=left, 1=right).
   RowOffset contains the offset in bytes from one scan line to the next,
   controlling whether the arc is drawn up or down. Length is the
   horizontal length in pixels of the arc, and DrawList is a list
   containing 0 for each point if the next point is horizontally aligned,
   and 1 if the next point is 1 pixel above or below diagonally */
void DrawHOctant(int X, int Y, int Length, int RowOffset,
   int HorizontalMoveDirection, unsigned char *DrawList)
{
   unsigned char far *ScreenPtr, BitMask;

   /* Point to the byte the initial pixel is in */
#ifdef __TURBOC__
   ScreenPtr = MK_FP(SCREEN_SEGMENT,
      (Y * SCREEN_WIDTH_IN_BYTES) + (X / 8));
#else
   FP_SEG(ScreenPtr) = SCREEN_SEGMENT;
   FP_OFF(ScreenPtr) =(Y * SCREEN_WIDTH_IN_BYTES) + (X / 8);
#endif
   /* Set the initial bit mask */
   BitMask = 0x80 >> (X & 0x07);

   /* Draw all points in DrawList */
   while ( Length-- ) {
      /* Set the bit mask for the pixel */
      outp(GC_INDEX + 1, BitMask);
      /* Draw the pixel (see comments above for details) */
      *ScreenPtr |= 0xFF;
      /* Now advance to the next pixel based on DrawList */
      if ( *DrawList++ ) {
         /* Advance vertically to produce a diagonal move */
         ScreenPtr += RowOffset; /* advance to the next scan line */
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
      }
      /* Advance horizontally. Rotate the bit mask, advancing one
         byte horizontally if the bit mask wraps */
      if ( HorizontalMoveDirection == 1 ) {
         /* Move right */
         if ( (BitMask >>= 1) == 0 ) {
            BitMask = 0x80;    /* wrap the mask */
            ScreenPtr++;       /* advance 1 byte to the right */
         }
      } else {
         /* Move left */
         if ( (BitMask <<= 1) == 0 ) {
            BitMask = 0x01;    /* wrap the mask */
            ScreenPtr--;       /* advance 1 byte to the left */
         }
      }
   }
}

/* Draws a circle of radius Radius in color Color centered at
 * screen coordinate (X,Y) */
void DrawCircle(int X, int Y, int Radius, int Color) {
   int MajorAxis, MinorAxis;
   unsigned long RadiusSqMinusMajorAxisSq, MinorAxisSquaredThreshold;
   unsigned char *PixListPtr;

   /* Set drawing color via set/reset */
   outpw(GC_INDEX, (0x0F << 8) | SET_RESET_ENABLE_INDEX);
                              /* enable set/reset for all planes */
   outpw(GC_INDEX, (Color << 8) | SET_RESET_INDEX);
                              /* set set/reset (drawing) color */
   outp(GC_INDEX, BIT_MASK_INDEX); /* leave the GC Index reg pointing
                                      to the Bit Mask reg */

   /* Set up to draw the circle by setting the initial point to one
      end of the 1/8th of a circle arc we'll draw */
   MajorAxis = 0;
   MinorAxis = Radius;
   /* Set initial Radius**2 - MajorAxis**2 (MajorAxis is initially 0) */
   RadiusSqMinusMajorAxisSq = (unsigned long) Radius * Radius;
   /* Set threshold for minor axis movement at (MinorAxis - 0.5)**2 */
   MinorAxisSquaredThreshold = (unsigned long) MinorAxis * MinorAxis -
      MinorAxis;

   /* Calculate all points along an arc of 1/8th of the circle and
      store that info in PixList for later drawing */
   PixListPtr = PixList;
   do {
      /* Advance (Radius**2 - MajorAxis**2); if it equals or passes
         the MinorAxis**2 threshold, advance one pixel along both the
         major and minor axes and set the next MinorAxis**2 threshold;
         otherwise, advance one pixel only along the major axis */
      RadiusSqMinusMajorAxisSq -=
         MajorAxis + MajorAxis + 1;
      if ( RadiusSqMinusMajorAxisSq <= MinorAxisSquaredThreshold ) {
         /* Advance 1 pixel along both the major and minor axes */
         MinorAxis--;
         MinorAxisSquaredThreshold -= MinorAxis + MinorAxis;
         *PixListPtr++ = 1;    /* advance along both axes */
      } else {
         *PixListPtr++ = 0;    /* advance only along the major axis */
      }
      MajorAxis++;   /* always advance one pixel along the major axis */
   } while ( MajorAxis <= MinorAxis );
```



```
   /* Now draw each of the 8 symmetries of the octant in turn */
   /* Draw the octants for which Y is the major axis */
   DrawVOctant(X-Radius,Y,MajorAxis,-SCREEN_WIDTH_IN_BYTES,1,PixList);
   DrawVOctant(X-Radius,Y,MajorAxis,SCREEN_WIDTH_IN_BYTES,1,PixList);
   DrawVOctant(X+Radius,Y,MajorAxis,-SCREEN_WIDTH_IN_BYTES,0,PixList);
   DrawVOctant(X+Radius,Y,MajorAxis,SCREEN_WIDTH_IN_BYTES,0,PixList);
   /* Draw the octants for which X is the major axis */
   DrawHOctant(X,Y-Radius,MajorAxis,SCREEN_WIDTH_IN_BYTES,0,PixList);
   DrawHOctant(X,Y-Radius,MajorAxis,SCREEN_WIDTH_IN_BYTES,1,PixList);
   DrawHOctant(X,Y+Radius,MajorAxis,-SCREEN_WIDTH_IN_BYTES,0,PixList);
   DrawHOctant(X,Y+Radius,MajorAxis,-SCREEN_WIDTH_IN_BYTES,1,PixList);

   /* Reset the Bit Mask register to normal */
   outp(GC_INDEX + 1, 0xFF);
   /* Turn off set/reset enable */
   outpw(GC_INDEX, (0x00 << 8) | SET_RESET_ENABLE_INDEX);
}
```

LISTING 2

```
/* Draws a series of concentric circles.
 * For VGA only, because mode 12h is unique to VGA.
 * Compile and link with accompanying listings as follows:
 *    tcc listing1 listing2                (Turbo C)
 *    tcc listing3 listing2 listing4.asm   (Turbo C)
 *    cl listing1.c listing2.c             (MSC)
 *    cl listing3.c listing2.c listing4;   (MSC; listing4.asm
 *                                         must already be assembled)
 */
#include <dos.h>

main() {
   int Radius, Temp, Color, i;
   union REGS Regs;

   /* Select VGA's hi-res 640x480 graphics mode, mode 12h */
   Regs.x.ax = 0x0012;
   int86(0x10, &Regs, &Regs);

   /* Draw 20 sets of concentric circles for timing purposes */
   for (i = 0; i < 20; i++) {
      for ( Radius = 10, Color = 7; Radius < 240; Radius += 2 ) {
         DrawCircle(640/2, 480/2, Radius, Color);
         Color = (Color + 1) & 0x0F;   /* cycle through 16 colors */
      }
   }

   /* Wait for a key, restore text mode, and done */
   scanf("%c", &Temp);
   Regs.x.ax = 0x0003;
   int86(0x10, &Regs, &Regs);
}
```

LISTING 3

```
/* Draws a circle of the specified radius and color, using a fast
 * integer-only & square-root-free approach, and generating the
 * arc for one octant into a buffer, then drawing all 8 symmetries
```

Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Abrash code, continued

5

```
 * from that buffer. Uses assembly language for inner loops of octant
 * generation & drawing.
 * Compiles with either Turbo C 2.0 or MSC 5.0.
 * Will work on VGA or EGA, but will draw what appears to be an
 * ellipse in non-square-pixel modes.
 */
#define ISVGA  0              /* set to 1 to use VGA write mode 3*/
                              /* keep synchronized with Listing 4 */
#include <dos.h>

/* Handle differences between Turbo C & MSC. Note that Turbo C accepts
   outp as a synonym for outportb, but not outpw for outport */
#ifdef __TURBOC__
#define outpw  outport
#endif

#define SCREEN_WIDTH_IN_BYTES 80    /* # of bytes across one scan
                                       line in mode 12h */
#define SCREEN_SEGMENT        0xA000 /* mode 12h display memory seg */
#define GC_INDEX              0x3CE /* Graphics Controller port */
#define SET_RESET_INDEX       0  /* Set/Reset reg index in GC */
#define SET_RESET_ENABLE_INDEX 1 /* Set/Reset Enable reg index in GC */
#define GC_MODE_INDEX         5  /* Graphics Mode reg index in GC */
#define COLOR_DONT_CARE       7  /* Color Don't Care reg index in GC */
#define BIT_MASK_INDEX        8  /* Bit Mask reg index in GC */

unsigned char PixList[SCREEN_WIDTH_IN_BYTES*8/2];
                                  /* maximum major axis length is
                                     1/2 screen width, because we're
                                     assuming no clipping is needed */
/* Draws a circle of radius Radius in color Color centered at
 * screen coordinate (X,Y) */
void DrawCircle(int X, int Y, int Radius, int Color) {
   int MajorAxis, MinorAxis;
   unsigned long RadiusSqMinusMajorAxisSq, MinorAxisSquaredThreshold;
   unsigned char *PixListPtr, OriginalGCMode;

   /* Set drawing color via set/reset */
   outpw(GC_INDEX, (0x0F << 8) | SET_RESET_ENABLE_INDEX);
                              /* enable set/reset for all planes */
   outpw(GC_INDEX, (Color << 8) | SET_RESET_INDEX);
                              /* set set/reset (drawing) color */
#if ISVGA
   /* Remember original read/write mode & select
      read mode 1/write mode 3, with Color Don't Care
      set to ignore all planes and therefore always return 0xFF */
   outp(GC_INDEX, GC_MODE_INDEX);
   OriginalGCMode = inp(GC_INDEX + 1);
   outp(GC_INDEX+1, OriginalGCMode | 0x0B);
   outpw(GC_INDEX, (0x00 << 8) | COLOR_DONT_CARE);
   outpw(GC_INDEX, (0xFF << 8) | BIT_MASK_INDEX);
#else
   outp(GC_INDEX, BIT_MASK_INDEX); /* leave the GC Index reg pointing
                                      to the Bit Mask reg */
#endif

   /* Set up to draw the circle by setting the initial point to one
      end of the 1/8th of a circle arc we'll draw */
   MajorAxis = 0;
   MinorAxis = Radius;
   /* Set initial Radius**2 - MajorAxis**2 (MajorAxis is initially 0) */
   RadiusSqMinusMajorAxisSq = (unsigned long) Radius * Radius;
   /* Set threshold for minor axis movement at (MinorAxis - 0.5)**2 */
   MinorAxisSquaredThreshold = (unsigned long) MinorAxis * MinorAxis -
```

6

```
      MinorAxis;

   /* Calculate all points along an arc of 1/8th of the circle.
      Results are placed in PixList */
   MajorAxis = GenerateOctant(PixList, MajorAxis, MinorAxis,
      RadiusSqMinusMajorAxisSq, MinorAxisSquaredThreshold);

   /* Now draw each of the 8 symmetries of the octant in turn */
   /* Draw the octants for which Y is the major axis */
   DrawVOctant(X-Radius,Y,MajorAxis,-SCREEN_WIDTH_IN_BYTES,1,PixList);
   DrawVOctant(X-Radius,Y,MajorAxis,SCREEN_WIDTH_IN_BYTES,1,PixList);
   DrawVOctant(X+Radius,Y,MajorAxis,-SCREEN_WIDTH_IN_BYTES,0,PixList);
   DrawVOctant(X+Radius,Y,MajorAxis,SCREEN_WIDTH_IN_BYTES,0,PixList);
   /* Draw the octants for which X is the major axis */
   DrawHOctant(X,Y-Radius,MajorAxis,SCREEN_WIDTH_IN_BYTES,0,PixList);
   DrawHOctant(X,Y-Radius,MajorAxis,SCREEN_WIDTH_IN_BYTES,1,PixList);
   DrawHOctant(X,Y+Radius,MajorAxis,-SCREEN_WIDTH_IN_BYTES,0, PixList);
   DrawHOctant(X,Y+Radius,MajorAxis,-SCREEN_WIDTH_IN_BYTES,1, PixList);

#if ISVGA
   /* Restore original write mode */
   outpw(GC_INDEX, (OriginalGCMode << 8) | GC_MODE_INDEX);
   /* Restore normal Color Don't Care setting */
   outpw(GC_INDEX, (0x0F << 8) | COLOR_DONT_CARE);
#else
   /* Reset the Bit Mask register to normal */
   outp(GC_INDEX + 1, 0xFF);
#endif
   /* Turn off set/reset enable */
   outpw(GC_INDEX, (0x00 << 8) | SET_RESET_ENABLE_INDEX);
}
```

LISTING 4

```
; Contains 3 C-callable routines: GenerateOctant, DrawVOctant, and
;       DrawHOctant. See individual routines for comments.
;
; Works with TASM 1.01 or MASM 5.1.
;
ISVGA   equ     0       ;set to 1 to use VGA write mode 3
                        ; keep synchronized with Listing 3
        .model small
        .code
;************************************************************************
; Generates an octant of the specified circle, placing the results in
; PixList, with a 0 in PixList meaning draw pixel & move only along major
; axis, and a 1 in PixList meaning draw pixel & move along both axes.
; C near-callable as:
;  int GenerateOctant(unsigned char *PixList, int MajorAxis,
;       int MinorAxis, unsigned long RadiusSqMinusMajorAxisSq,
;       unsigned long MinorAxisSquaredThreshold);
;
; Return value = MajorAxis
;
GenerateOctantParms     struc
        dw      ?       ;pushed BP
        dw      ?       ;return address
PixList dw      ?       ;pointer to list to store draw control data in
MajorAxis dw    ?       ;initial major/minor axis coords relative to
MinorAxis dw    ?       ; to the center of the circle
RadiusSqMinusMajorAxisSq dd ?  ;initial Radius**2 - MajorAxis**2
MinorAxisSquaredThreshold dd ? ;initial threshhold for minor axis
```

Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PRINTER GRAPHICS IN HIGH RESOLUTION!

GRAPHLINK™ Printer Graphics Toolkit

GraphLink™ lets your programs tap the *full potential* of printers to produce dazzling *printed* graphics in *high-resolution!* Compare the features ...

- Uses the *PRINTER'S* resolution, not the SCREEN'S!
- Can produce 300 dpi images on laser printers
- A linkable library of 80+ routines for building and printing images
- Provides line, bar, arc, pie-slice, pattern, and bitblt routines, fully-scalable fonts and much more!
- Works with any screen graphics software
- Emulates the Borland Graphics Interface (great for BGI programs!)
- *Can automatically* build the printed image, by intercepting your screen graphics calls (Turbo Pascal version)
- Supports the most popular laser and dot matrix printers
- Supports Turbo Pascal 4.0 and higher. Available soon for Turbo C, Microsoft C, and Quick C
- Only *$125* for the single-user version, or *$250* for GraphLink *Professional,* which includes an unlimited distribution license. (Shipping & Handling: USA $5, Canada $10, Other $15.)

30 Day No Risk Money-Back Guarantee

GRAPHLINK
The Printer Graphics Solution™

VISITECH SOFTWARE
D5 3807 Ridgewood Court
Pittsburgh, PA 15239 USA
(412) 733-4775

Number **304** on the Reader Service Card

Develop Powerful Image-based Applications Quickly and Easily

Introducing VICTOR, video capture and image processing library

Support economical frame grabbers
Versions of Victor are available for inexpensive 256 x 256 64-level frame grabbers as well as full-featured boards at 512 x 512 256-level resolution.

Add live video display and image processing features to your application
With Victor your applications can show live video on any portion of the VGA screen and combine live video with text and graphics. Use sophisticated image processing tools, create lineart, TIFF/PCX files, more. Victor handles all the low level functions so you can concentrate on your application.

Combine live video, text, and graphics on the VGA screen.

Rapidly test and prototype
Victor comes with ZIP Image Processing software to get you up and running quickly -- test and prototype functions before you write any code.

Victor supports Microsoft C v 5 and Quick C, and includes over 100 functions, demonstration and prototyping software, full documentation, and source code for device control routines . . . all for only $195.

Use low cost video digitizers to develop high powered applications.

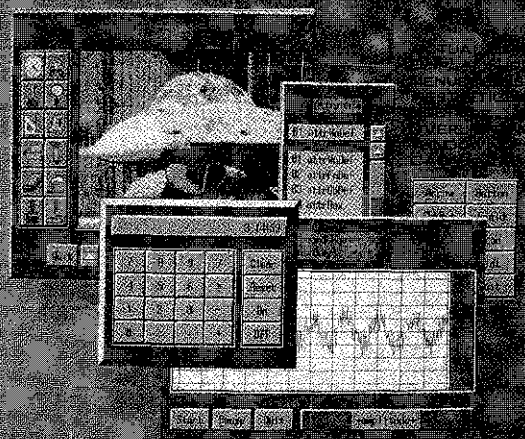
video frame grabbers available
Call (314) 962-7833 to order
VISA/MC/COD

Catenary Systems 470 Belleview St Louis MO 63119 (314) 962-7833

Number **321** on the Reader Service Card

[illegible]

[illegible]



[illegible]

Call for your free demo:
303.494.8863

ITHACA STREET SOFTWARE

1145 ITHACA DRIVE, BOULDER, CO, 80303, USA

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Abrash code, continued

7

```
GenerateOctantParms     ends    ; movement is MinorAxis**2 - MinorAxis
;
        public _GenerateOctant
_GenerateOctant proc    near
        push    bp              ;preserve caller's stack frame
        mov     bp,sp           ;point to our stack frame
        push    si              ;preserve C register variables
        push    di
;                               ;get all parms into registers
        mov     di,[PixList+bp]         ;point DI to PixList
        mov     ax,[MajorAxis+bp]       ;AX=MajorAxis
        mov     bx,[MinorAxis+bp]       ;BX=MinorAxis
        mov     cx,word ptr [RadiusSqMinusMajorAxisSq+bp]
        mov     dx,word ptr [RadiusSqMinusMajorAxisSq+bp+2]
                                ;DX:CX=RadiusSqMinusMajorAxisSq
        mov     si,word ptr [MinorAxisSquaredThreshold+bp]
        mov     bp,word ptr [MinorAxisSquaredThreshold+bp+2]
                                ;BP:SI=MinorAxisSquaredThreshold
GenLoop:
        sub     cx,1            ;subtract MajorAxis + MajorAxis + 1 from
        sbb     dx,0            ; RadiusSqMinusMajorAxisSq
        sub     cx,ax
        sbb     dx,0
        sub     cx,ax
        sbb     dx,0
        cmp     dx,bp           ;if RadiusSqMinusMajorAxisSq <=
        jb      IsMinorMove     ; MinorAxisSquaredThreshold, move along
        ja      NoMinorMove     ; minor as well as major, otherwise move
        cmp     cx,si           ; only along major
        ja      NoMinorMove
IsMinorMove:                    ;move along minor as well as major
        dec     bx              ;decrement MinorAxis
        sub     si,bx           ;subtract MinorAxis + MinorAxis from
        sbb     bp,0            ; MinorAxisSquaredThreshold
        sub     si,bx
        sbb     bp,0
        mov     byte ptr [di],1 ;enter 1 (move both axes) in PixList
        inc     di              ;advance PixList pointer
        inc     ax              ;increment MajorAxis
        cmp     ax,bx           ;done if MajorAxis > MinorAxis, else
        jbe     GenLoop         ; continue generating PixList entries
        jmp     short Done
NoMinorMove:
        mov     byte ptr [di],0 ;enter 0 (move only major) in PixList
        inc     di              ;advance PixList pointer
        inc     ax              ;increment MajorAxis
        cmp     ax,bx           ;done if MajorAxis > MinorAxis, else
        jbe     GenLoop         ; continue generating PixList entries
Done:
        pop     di              ;restore C register variables
        pop     si
        pop     bp
        ret
_GenerateOctant endp
;****************************************************************************
; Draws the arc for an octant in which Y is the major axis. (X,Y) is the
; starting point of the arc. HorizontalMoveDirection selects whether the
; arc advances to the left or right horizontally (0=left, 1=right).
; RowOffset contains the offset in bytes from one scan line to the next,
; controlling whether the arc is drawn up or down. DrawLength is the
; vertical length in pixels of the arc, and DrawList is a list
; containing 0 for each point if the next point is vertically aligned,
; and 1 if the next point is 1 pixel diagonally to the left or right.
;
```

8

```
; The Graphics Controller Index register must already point to the Bit
; Mask register.
;
; C near-callable as:
;  void DrawVOctant(int X, int Y, int DrawLength, int RowOffset,
;       int HorizontalMoveDirection, unsigned char *DrawList);
;
DrawParms       struc
        dw      ?       ;pushed BP
        dw      ?       ;return address
X       dw      ?       ;initial coordinates
Y       dw      ?
DrawLength dw   ?       ;vertical length
RowOffset dw    ?       ;distance from one scan line to the next
HorizontalMoveDirection dw ? ;1 to move right, 0 to move left
DrawList dw     ?       ;pointer to list containing 1 to draw
DrawParms       ends    ; diagonally, 0 to draw vertically for
                                ; each point
SCREEN_SEGMENT  equ     0a000h  ;display memory segment in mode 12h
SCREEN_WIDTH_IN_BYTES equ 80    ;distance from one scan line to next
GC_INDEX        equ     3ceh    ;GC Index register address
;
        public _DrawVOctant
_DrawVOctant    proc    near
        push    bp              ;preserve caller's stack frame
        mov     bp,sp           ;point to our stack frame
        push    si              ;preserve C register variables
        push    di
;Point ES:DI to the byte the initial pixel is in.
        mov     ax,SCREEN_SEGMENT
        mov     es,ax
        mov     ax,SCREEN_WIDTH_IN_BYTES
        mul     [bp+Y]  ;Y*SCREEN_WIDTH_IN_BYTES
        mov     di,[bp+X] ;X
        mov     cx,di   ;set X aside in CX
        shr     di,1
        shr     di,1
        shr     di,1    ;X/8
        add     di,ax   ;screen offset = Y*SCREEN_WIDTH_IN_BYTES+X/8
        and     cl,07h  ;X modulo 8
if ISVGA                ;---VGA---
        mov     ah,80h  ;keep VGA bit mask in AH
        shr     ah,cl   ;initial bit mask = 80h shr (X modulo 8);
        cld             ;for LODSB, used below
else                    ;---EGA---
        mov     al,80h  ;keep EGA bit mask in AL
        shr     al,cl   ;initial bit mask = 80h shr (X modulo 8);
        mov     dx,GC_INDEX+1 ;point DX to GC Data reg/bit mask
endif                   ;---------
        mov     si,[bp+DrawList] ;SI points to list to draw from
        sub     bx,bx           ;so we have the constant 0 in a reg
        mov     cx,[bp+DrawLength] ;CX=# of pixels to draw
        jcxz    VDrawDone       ;skip this if no pixels to draw
        cmp     [bp+HorizontalMoveDirection],0 ;draw right or left
        mov     bp,[bp+RowOffset] ;BP=offset to next row
        jz      VGoLeft         ;draw from right to left
VDrawRightLoop:                 ;draw from left to right
if ISVGA                        ;---VGA---
        and     es:[di],ah      ;AH becomes bit mask in write mode 3,
                                ; set/reset provides color
        lodsb                   ;get next draw control byte
        and     al,al           ;move right?
        jz      VAdvanceOneLineRight ;no move right
        ror     ah,1            ;move right
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
else                                    ;---EGA---
        out     dx,al                   ;set the desired bit mask
        and     es:[di],al              ;data doesn't matter (set/reset provides
                                        ; color); just force read then write
        cmp     [si],bl                 ;check draw control byte; move right?
        jz      VAdvanceOneLineRight ;no move right
        ror     al,1                    ;move right
endif                                   ;---------
        adc     di,bx   ;move one byte to the right if mask wrapped
VAdvanceOneLineRight:
ife ISVGA                               ;---EGA---
        inc     si                      ;advance draw control list pointer
endif                                   ;---------
        add     di,bp                   ;move to the next scan line up or down
        loop    VDrawRightLoop  ;do next pixel, if any
        jmp     short VDrawDone ;done
VGoLeft:                                ;draw from right to left
VDrawLeftLoop:
if ISVGA                                ;---VGA---
        and     es:[di],ah              ;AH becomes bit mask in write mode 3
        lodsb                           ;get next draw control byte
        and     al,al                   ;move left?
        jz      VAdvanceOneLineLeft ;no move left
        rol     ah,1                    ;move left
else                                    ;---EGA---
        out     dx,al                   ;set the desired bit mask
        and     es:[di],al              ;data doesn't matter; force read/write
        cmp     [si],bl                 ;check draw control byte; move left?
        jz      VAdvanceOneLineLeft ;no move left
        rol     al,1                    ;move left
endif                                   ;---------
        sbb     di,bx                   ;move one byte to the left if mask wrapped
VAdvanceOneLineLeft:
ife ISVGA                               ;---EGA---
        inc     si                      ;advance draw control list pointer
endif                                   ;---------
        add     di,bp                   ;move to the next scan line up or down
        loop    VDrawLeftLoop   ;do next pixel, if any
VDrawDone:
        pop     di                      ;restore C register variables
        pop     si
        pop     bp
        ret
_DrawVOctant    endp
;*************************************************************************
; Draws the arc for an octant in which X is the major axis. (X,Y) is the
; starting point of the arc. HorizontalMoveDirection selects whether the
; arc advances to the left or right horizontally (0=left, 1=right).
; RowOffset contains the offset in bytes from one scan line to the next,
; controlling whether the arc is drawn up or down. DrawLength is the
; horizontal length in pixels of the arc, and DrawList is a list
; containing 0 for each point if the next point is horizontally aligned,
; and 1 if the next point is 1 pixel above or below diagonally.
;
; Graphics Controller Index register must already point to the Bit Mask
; register.
;
; C near-callable as:
;  void DrawHOctant(int X, int Y, int DrawLength, int RowOffset,
;       int HorizontalMoveDirection, unsigned char *DrawList)
;
; Uses same parameter structure as DrawVOctant().
;
        public _DrawHOctant
```

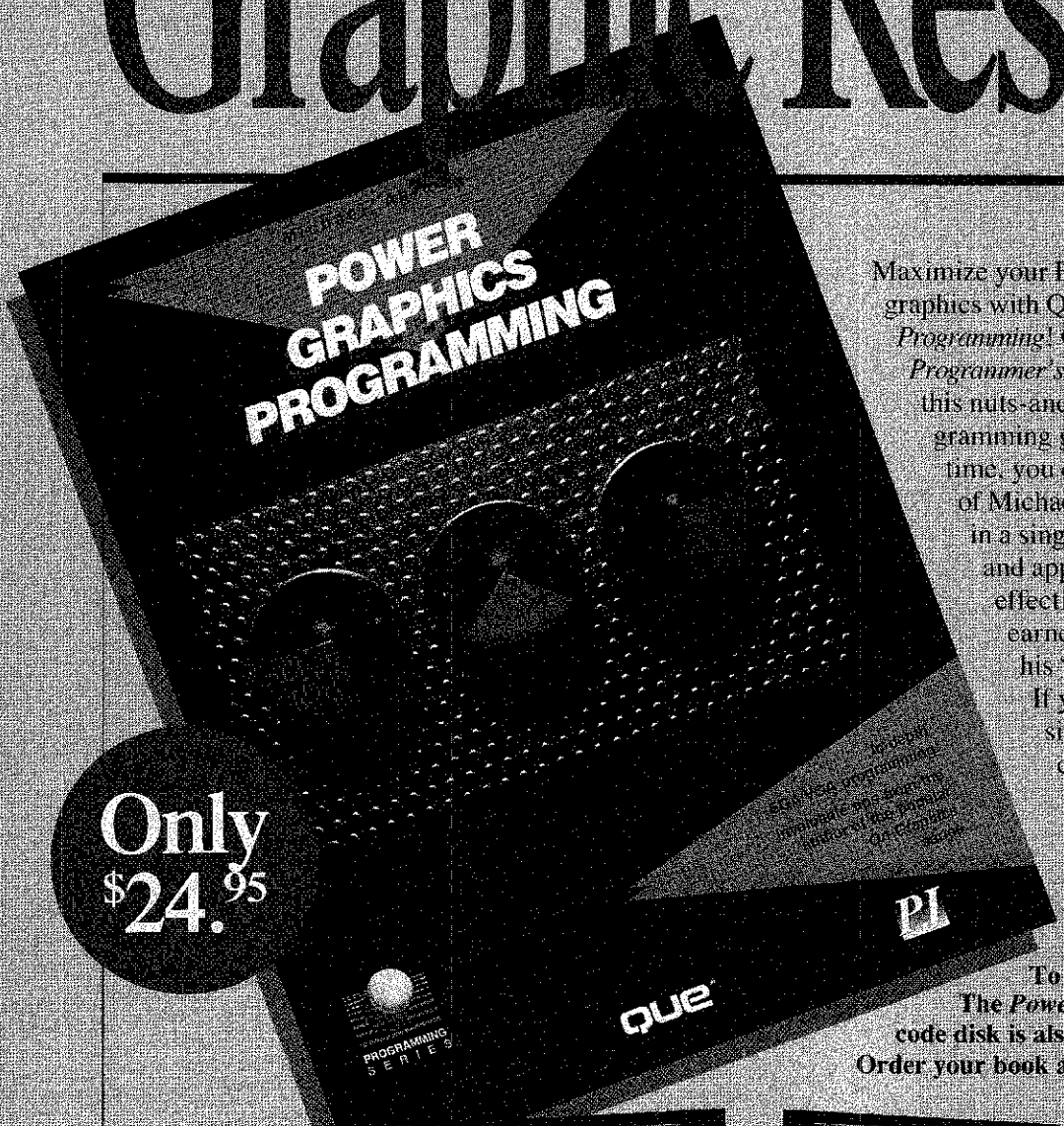


```
_DrawHOctant    proc    near
        push    bp              ;preserve caller's stack frame
        mov     bp,sp           ;point to our stack frame
        push    si              ;preserve C register variables
        push    di
;Point ES:DI to the byte the initial pixel is in.
        mov     ax,SCREEN_SEGMENT
        mov     es,ax
        mov     ax,SCREEN_WIDTH_IN_BYTES
        mul     [bp+Y]  ;Y*SCREEN_WIDTH_IN_BYTES
        mov     di,[bp+X] ;X
        mov     cx,di   ;set X aside in CX
        shr     di,1
        shr     di,1
        shr     di,1    ;X/8
        add     di,ax   ;screen offset = Y*SCREEN_WIDTH_IN_BYTES+X/8
        and     cl,07h  ;X modulo 8
        mov     bh,80h
        shr     bh,cl   ;initial bit mask = 80h shr (X modulo 8);
if ISVGA                ;---VGA---
        cld             ;for LODSB, used below
else                    ;---EGA---
        mov     dx,GC_INDEX+1 ;point DX to GC Data reg/bit mask
endif                   ;---------
        mov     si,[bp+DrawList] ;SI points to list to draw from
        sub     bl,bl           ;so we have the constant 0 in a reg
        mov     cx,[bp+DrawLength] ;CX=# of pixels to draw
        jcxz    HDrawDone       ;skip this if no pixels to draw
if ISVGA                        ;---VGA---
        sub     ah,ah           ;clear bit mask accumulator
else                            ;---EGA---
        sub     al,al           ;clear bit mask accumulator
endif                           ;---------
        cmp     [bp+HorizontalMoveDirection],0 ;draw right or left
        mov     bp,[bp+RowOffset] ;BP=offset to next row
        jz      HGoLeft         ;draw from right to left
HDrawRightLoop:                 ;draw from left to right
if ISVGA                        ;---VGA---
        or      ah,bh           ;put this pixel in bit mask accumulator
        lodsb                   ;get next draw control byte
        and     al,al           ;move up/down?
else                            ;---EGA---
        or      al,bh           ;put this pixel in bit mask accumulator
        cmp     [si],bl         ;check draw control byte; move up/down?
endif                           ;---------
        jz      HAdvanceOneLineRight ;no move up/down
                        ;move up/down; first draw accumulated pixels
if ISVGA                        ;---VGA---
        and     es:[di],ah      ;AH becomes bit mask in write mode 3
        sub     ah,ah           ;clear bit mask accumulator
else                            ;---EGA---
        out     dx,al           ;set the desired bit mask
        and     es:[di],al      ;data doesn't matter; force read/write
        sub     al,al           ;clear bit mask accumulator
endif                           ;---------
        add     di,bp           ;move to the next scan line up or down
HAdvanceOneLineRight:
ife ISVGA                       ;---EGA---
        inc     si              ;advance draw control list pointer
endif                           ;---------
        ror     bh,1            ;move to right; shift mask
        jnc     HDrawLoopRightBottom ;didn't wrap to the next byte
                        ;move to next byte; 1st draw accumulated pixels
if ISVGA                        ;---VGA---
```

Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Graphic Results!

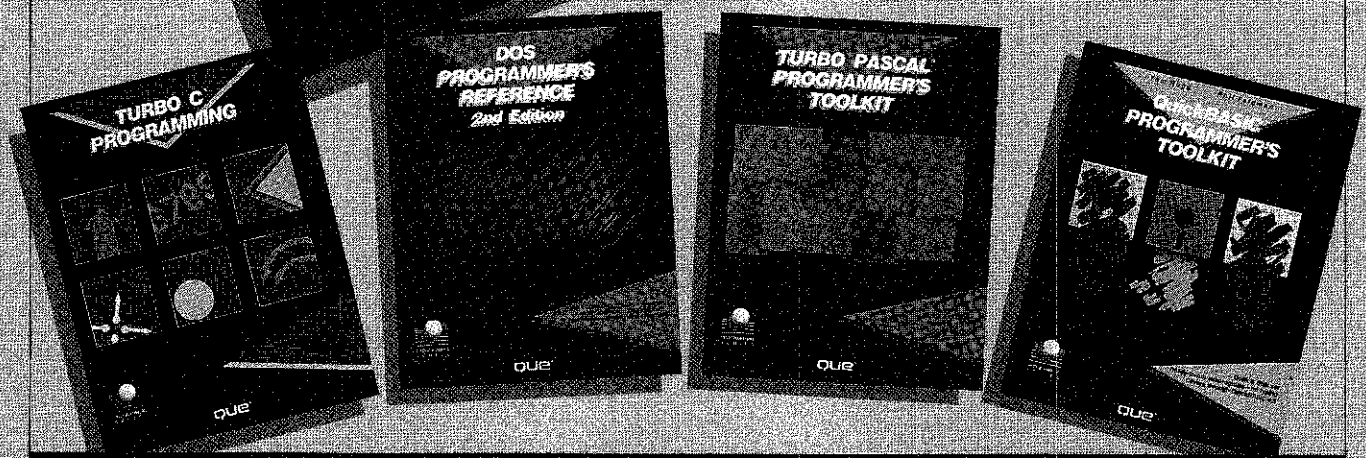


Only $24.95

Maximize your EGA/VGA programming graphics with Que's *Power Graphics Programming*! Que has teamed up with *Programmer's Journal* to bring you this nuts-and-bolts graphics programming guide. Now, for the first time, you can have the expertise of Michael Abrash consolidated in a single volume. His insights and approach to efficient, effective programming have earned him high acclaim for his "On Graphics" series. If you are seeking that elusive graphic edge, you can't be without *Power Graphics Programming*!

Order yours directly from *Programmer's Journal*

To order call 800-234-0386. The *Power Graphics Programming* code disk is also available for $14.95. Order your book and code disk today!



Look for these other practical Programming books from Que

Available in better bookstores and computer stores everywhere. Or call 800-428-5331, ext. PJ01.

que®

Number 296 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Abrash code, continued

```
        and     es:[di],ah      ;AH becomes bit mask in write mode 3
        sub     ah,ah           ;clear bit mask accumulator
    else
        out     dx,al           ;set the desired bit mask
        and     es:[di],al      ;data doesn't matter; force read/write
        sub     al,al           ;clear bit mask accumulator
    endif
        inc     di              ;move 1 byte to the right
HDrawLoopRightBottom:
        loop    HDrawRightLoop  ;draw next pixel, if any
        jmp     short HDrawDone ;done
HGoLeft:                        ;draw from right to left
HDrawLeftLoop:
    if ISVGA
        or      ah,bh           ;---VGA---
        lodsb                   ;put this pixel in bit mask accumulator
        and     al,al           ;get next draw control byte
                                ;move up/down?
    else                        ;---EGA---
        or      al,bh           ;put this pixel in bit mask accumulator
        cmp     [si],bl         ;check draw control byte; move up/down?
    endif                       ;--------
        jz      HAdvanceOneLineLeft ;no move up/down
                                ;move up/down; first draw accumulated pixels
    if ISVGA                    ;---VGA---
        and     es:[di],ah      ;AH becomes bit mask in write mode 3
        sub     ah,ah           ;clear bit mask accumulator
    else                        ;---EGA---
        out     dx,al           ;set the desired bit mask
        and     es:[di],al      ;data doesn't matter; force read/write
        sub     al,al           ;clear bit mask accumulator
    endif                       ;--------
        add     di,bp           ;move to the next scan line up or down
HAdvanceOneLineLeft:
    ife ISVGA                   ;---EGA---
        inc     si              ;advance draw control list pointer
    endif                       ;--------
        rol     bh,1            ;move to left; shift mask
        jnc     HDrawLoopLeftBottom ;didn't wrap to next byte
                                ;move to next byte; 1st draw accumulated pixels
    if ISVGA                    ;---VGA---
        and     es:[di],ah      ;AH becomes bit mask in write mode 3
        sub     ah,ah           ;clear bit mask accumulator
    else                        ;---EGA---
        out     dx,al           ;set the desired bit mask
        and     es:[di],al      ;data doesn't matter; force read/write
        sub     al,al           ;clear bit mask accumulator
    endif                       ;--------
        dec     di              ;move 1 byte to the left
HDrawLoopLeftBottom:
        loop    HDrawLeftLoop   ;draw next pixel, if any
HDrawDone:                      ;draw any remaining accumulated pixels
    if ISVGA                    ;---VGA---
        and     es:[di],ah      ;AH becomes bit mask in write mode 3
    else                        ;---EGA---
        out     dx,al           ;set the desired bit mask
        and     es:[di],al      ;data doesn't matter; force read/write
    endif                       ;--------
        pop     di              ;restore C register variables
        pop     si
        pop     bp
        ret
_DrawHOctant    endp
        end
```


Improved Atom Manager for Microsoft Windows™

THE ATOMIZER™

The **ATOMIZER** is an atom management library that extends capability of the Windows atom management facility. The extended features, which can be enabled and disabled at runtime, provide:

- An arbitrary number of atoms (up to 16384)
- Strings of arbitrary length (up to 65535)
- Use of empty string (" ")
- Explicit access to the atom reference count
- Separate atom assignment of strings differing only in case

The **ATOMIZER** library is strictly compatible with Windows global and local atom managers and can be retrofitted to existing applications without changing the application source code. **ATOMIZER** minimizes *usage of global and local memory with sophisticated* storage management algorithms.

The basic package contains the **ATOMIZER** dynamic link library and user-level documentation, with instructions and examples. The examples are supplied with their source files and auxiliary files needed to build example applications. In addition, the source package contains the library source code and an extensive program logic manual embedded via commentary in the source code.

Basic Package	3 1/2 or 5 1/4 diskette	**$39.50**
Source Package	3 1/2 or 5 1/4 diskette	**$79.50**

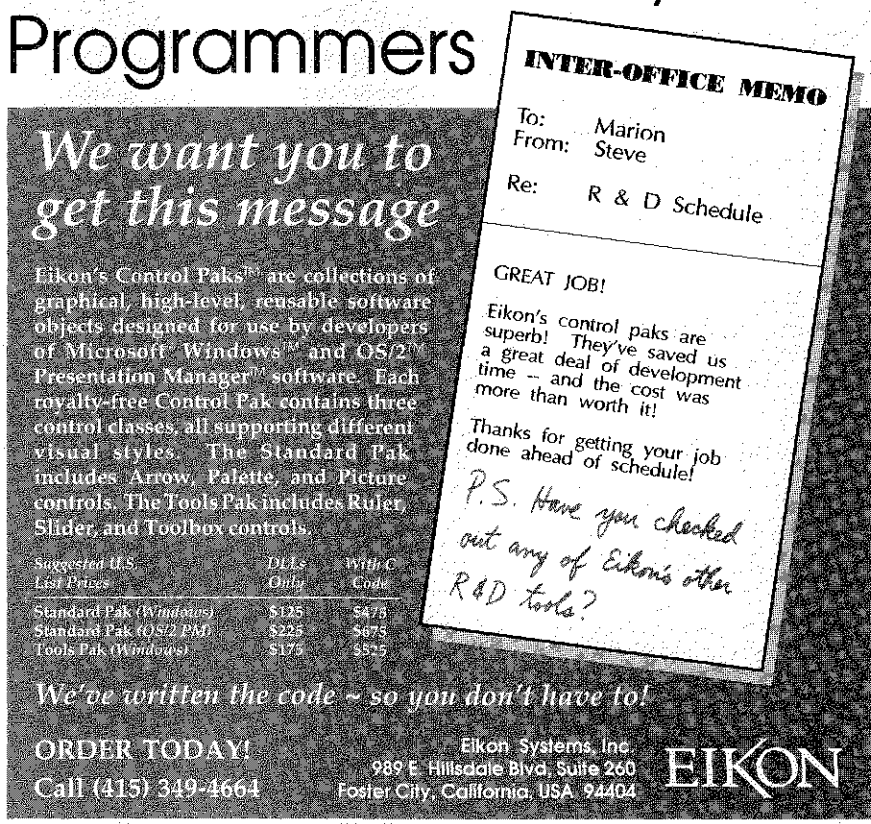
Please specify package and diskette size when ordering. Shipping and handling included; COD orders add $5.00. Send check or money order to:

Software Projects Group
3451 Phoebe Court
West Covina, CA 91792
(818) 913-7170

Trademark: ATOMIZER, Software Projects Group

Number **356** on the Reader Service Card

Windows and OS/2 PM Programmers


We want you to get this message
Eikon's Control Paks™ are collections of graphical, high-level, reusable software objects designed for use by developers of Microsoft Windows™ and OS/2™ Presentation Manager™ software. Each royalty-free Control Pak contains three control classes, all supporting different visual styles. The Standard Pak includes Arrow, Palette, and Picture controls. The Tools Pak includes Ruler, Slider, and Toolbox controls.
Suggested US List Prices
DLLs Only
With C Code
Standard Pak (Windows) $195 $475
Standard Pak (OS/2 PM) $225 $575
Tools Pak (Windows) $175 $525
INTER-OFFICE MEMO
To: Marion
From: Steve
Re: R & D Schedule
GREAT JOB!
Eikon's control paks are superb! They've saved us a great deal of development time -- and the cost was more than worth it!
Thanks for getting your job done ahead of schedule!
P.S. Have you checked out any of Eikon's other R&D tools?
We've written the code ~ so you don't have to!
ORDER TODAY!
Call (415) 349-4664
Eikon Systems, Inc.
989 E. Hillsdale Blvd, Suite 260
Foster City, California, USA 94404
EIKON

Number **349** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Put a TIGA in Your PC

by
Jake Richter

First, a little history . . .

Just over a year ago, I expounded in this magazine on the virtues of using a graphics processor to enhance graphics performance on your PC (*PJ* 7.1, January/February 1989). In that article, I discussed the Intel 82786 and the TI 34010. Intel, sad to say, has announced that it is dropping the 82786; the last chips will roll off the product line in mid 1991. The 34010, meanwhile, has become something of a hot ticket. The 34010 GSP (Graphic System Processor) is currently used in more than a hundred different products, primarily graphics boards.

While TI's graphics-board success with the 34010 was remarkable for the high-end graphics business, it was not great compared with the popularity of something more mainstream, such as the VGA. There were two major features missing: low cost and compatibility. TI realized that they had to take some drastic actions to penetrate the low-end graphics market.

Lowering the price of 34010-based products has been taken care of by some clever marketing, price cutting, and consolidation of technology. The latter is the TI 34092 gate array (also known as the Business Graphics Array, or BGA), which combines most of the necessary chip-level logic for a 34010 board onto one piece of silicon, potentially reducing the overall chip count for a complete 34010 board to 16 chips, including the RAM. Thanks to the 34092 and dropping RAM prices, you should see 34010 boards retailing for under $600 soon.

The lack of compatibility was really a matter of incompatibility, not with existing PC graphics, but among the sundry 34010-based graphics boards. There are various graphics-company-induced interface standards: Renaissance has the RGDI interface, VMI (Vermont Microsystems, Inc.) has PGL, Number Nine has NNIOS, and so on. The problem was that none of these interfaces had a substantial following in the marketplace, and while the interfaces were robust, there was some concern about their performance. In deciding to standardize 34010 graphics, TI reviewed all the interfaces, chose the best features, added some Texan pixie dust, and *voilà!*, TIGA was born.

TIGA, which stands for Texas Instruments Graphics Architecture, has the benefit of TI's size and marketing capital behind it. By the time you read this, at least two dozen graphics-board companies will be advertising TIGA support, and many major PC graphics applications will also support the TIGA interface in one way or another. TIGA will be a standard to contend with, as well as one to support if you are serious about PC graphics.

Inside TIGA

The engineers at TI, to my knowledge, had four design goals, which if achieved would make TIGA a standard:

- simplicity in porting
- ease of use
- extensibility
- high performance

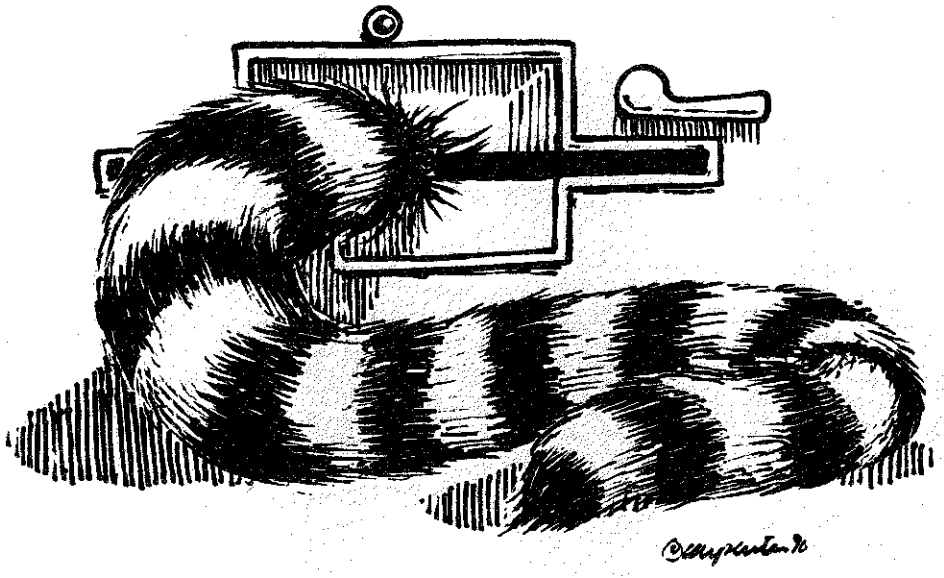
Let's look at the structure of TIGA and then see how these four goals were met.

TIGA consists primarily of two portions: a PC-resident portion, called the "Communications Driver," and a TI 34010–resident portion, called the "Graphics Manager."

Communications Driver

TIGA's Communications Driver, or CD, controls the communications between the graphics board and the PC and supports a number of functions that allow you transfer data to and from the board in different ways. The CD is normally the only portion of TIGA that needs to be modified when porting to a new graphics board. The Application Interface, or AI (*not* to be confused with IBM's 8514/A Adapter Interface), is a function library that translates PC-based TIGA commands into a format that the CD can use. The AI communicates with the CD via an interrupt, usually INT 7FH. The interrupt number can be altered

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



dynamically by way of an environment variable.

The AI is provided mainly to simplify programming via TIGA; it consists of several C and assembly language include files and a library to be linked into your program. (You could write your own AI, but I wouldn't recommend it, as TI has not documented the interface to the CD.) The include files contain information about TIGA structures and macro definitions that convert TIGA function names and parameters into a smaller set of functions, most of which use a special token–packet combination for transferring information. Unless you are writing custom-board-resident code to run with TIGA, you don't really need to know how the macros work; you just need to make a normal C function call, as listed in the TIGA documentation.

You install the CD by running a program called TIGACD, which is supplied with the graphics board. TIGACD searches the environment for the TIGA environment variable, which tells the CD where to find the standard TIGA files, where to find the user-defined uploadable code, and what interrupt to use for communicating with an application. Once TIGACD has all this information, it takes care of some other internal initializations and installs itself as a terminate-and-stay-resident program (TSR).

Graphics Manager

The Graphics Manager, or GM, is the TIGA code resident on the TI 34010–based graphics board. This board-resident code is responsible for initialization of the board, board memory management, the board side of the communications, and drawing. The GM comprises two distinct parts: the core primitives and the extended primitives.

The core primitives consist of functions that must be present to run TIGA. These include communications functions, some of which are actually implemented in the CD, such as core-graphics functions for setting colors, setting drawing modes, and manipulating the cursor. Once the GM is loaded, the core primitives are available. TI has integrated the GM load operation into the CD. This works well, since one of the first things a TIGA application usually

TIGA Tools

There are three different levels of software for TIGA development.

At the low end, you have the TIGA Driver Developer's Kit (DDK). The DDK contains everything you need to in order to write PC based TIGA applications and drivers (assuming you have a MASM v5.0 compatible assembler and/or Microsoft C v5.1). Contained on the DDK diskettes are dozens of sample programs which utilize TIGA both from C and assembly language, including a sample AutoCAD Release 9 driver for TIGA. The DDK sells for $340.

In the middle, you have the TMS340 Software Developer's Kit (SDK). The SDK contains the tools necessary for developing 340X0-board-based code on a PC, including TIGA downloadable extensions. The SDK includes a 340X0 C compiler, assembler, simulator (uses the CGA), librarian, a graphics and math library, fonts, and linker. Object code produced by the tools is in the Common Object File Format (COFF) used by many UNIX tools. The SDK includes the DDK, and retails for $1495.

At the high end is the TIGA Software Porting Kit (SPK). The SPK includes both the DDK and SDK, as well as the source code to all of TIGA and test suites to test the port. The source code allows board manufacturers to port TIGA to their 340X0 based boards. Also included is a Microsoft Windows driver for TIGA, which has probably been the one of the biggest incentives for many board manufacturers to go with TIGA. Windows drivers for graphics processor based boards are very painful to develop with any reasonable amount of success, and with the growing popularity of MS-Windows, it's a driver that graphics boards can't be shipped without. The SPK sells for $15,000, which includes a license to ship TIGA with the company's graphics boards.

To get more information on how to order the TIGA tools, as well as a free copy of the TIGA-340 Interface User's Guide, call TI at 1-800-336-5236 x3526 or try the TI Graphics Hot Line at 713-274-2340.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

TIGA Primitives

Function	Description	Type
bitblt	Bitblt source array to destination	Extended
cd_is_alive	Return if TIGACD is running	Core
clear_frame_buffer	Clear entire frame buffer	Core
clear_page	Clear current drawing page	Core
clear_screen	Clear screen	Core
cop2gsp	Copy coprocessor to GSP memory	Core
cpw	Compare point to window	Core
create_alm	Create absolute load module	Core
create_esym	Create external symbol table file	Core
delete_font	Remove a font from the font table	Extended
draw_line	Draw line	Extended
draw_oval	Draw ellipse outline	Extended
draw_ovalarc	Draw ellipse arc	Extended
draw_piearc	Draw ellipse pie slice	Extended
draw_point	Draw single pixel	Extended
draw_polyline	Draw list of lines	Extended
draw_polyline	Draw polyline	Extended
draw_rect	Draw rectangle outline	Extended
field_extract	Extract data from GSP memory	Core
field_insert	Insert data into GSP memory	Core
fill_convex	Draw solid convex polygon	Extended
fill_convex	Fill convex polygon	Extended
fill_oval	Draw solid ellipse	Extended
fill_piearc	Draw solid ellipse pie slice	Extended
fill_polygon	Draw solid polygon	Extended
fill_polygon	Fill polygon	Extended
fill_polygon	Fill polygon	Extended
fill_rect	Draw solid rectangle	Extended
flush_esym	Flush external symbol table file	Core
flush_extended	Flush all user extensions	Core
frame_oval	Draw oval border	Extended
frame_rect	Draw rectangular border	Extended
function_implemented	Return if a function is implemented	Core
get_colors	Return foreground and background colors	Core
get_config	Return board configuration	Core
get_curs_state	Return cursor current state	Core
get_curs_xy	Return cursor position	Core
get_env	Returns current environment structure	Core
get_fontinfo	Return font physical information	Core
get_isr_priorities	Return interrupt service routine priorities	Core
get_modeinfo	Return board configuration	Core
get_nearest_color	Return nearest color in a palette	Core
get_offscreen_memory	Return offscreen memory blocks	Core
get_palet	Return an entire palette	Core
get_palet_entry	Return a palette entry	Core
get_pixel	Read contents of a pixel	Extended
get_pmask	Returns color plane mask	Core
get_ppop	Returns pixel processing operation	Core
get_textattr	Return text rendering attributes	Extended
get_transp	Returns transparency mode	Core
get_vector	Get address at a TMS340 trap vector	Core
get_videomode	Return current emulation mode	Core
get_windowing	Inquire windowing mode	Core
get_wksp	Return offscreen workspace	Core
gsp_calloc	Allocate and clear GSP memory	Core
gsp_execute	Execute a COFF program	Core
gsp_free	Deallocate GSP memory	Core
gsp_malloc	Allocate GSP memory	Core
gsp_maxheap	Return largest free block	Core
gsp_minit	Reinitialize GSP memory heap pool	Core
gsp_realloc	Resize allocated block of memory	Core
gsp2cop	Copy GSP memory to coprocessor	Core
gsp2gsp	Copy from GSP Memory to GSP Memory	Core
gsp2host	Copy from GSP into host memory	Core
gsp2hostxy	Copy rectangular area from GSP to host	Core
host2gsp	Copy from host into GSP memory	Core
host2gspxy	Copy rectangular area from host to GSP	Core
init_palet	Default palette	Core
init_text	Initialize text drawing environment	Core

Continued . . .

does is switch into the TIGA display mode via the **set_videomode** command, and this causes the CD to load the GM if the GM isn't already loaded on the board.

The extended primitives have to be explicitly loaded with the **install_primitives** function. This requirement conserves the amount of graphics-board RAM used by TIGA. If applications are memory-usage sensitive, you can load only the extended primitives that are needed by using the TI 340X0 programming tools in the Software Developer's Kit (see the sidebar).

The names of all the TIGA primitives, their functions, and their types (core or extended) are provided in the list accompanying this article.

As you might gather, there are quite a number of functions that TIGA can perform for you without requiring you to write extra board-resident code. Let's take a look at a simple TIGA program that puts up a bunch of rectangles, lines, and text. The code for this program is provided in Listing 1.

TIGA's coordinate system works like that of most other graphics boards, namely with the origin in the upper left corner and *X* and *Y* increasing right and down, respectively. As you can see from the example, it is fairly easy to write a program that utilizes TIGA calls and produces something displayable at any resolution. Using the large range of functions that TIGA provides will satisfy most needs, but if you are not completely satisfied until you are manipulating raw bits, TIGA offers the ability to upload custom GSP code to the graphics board.

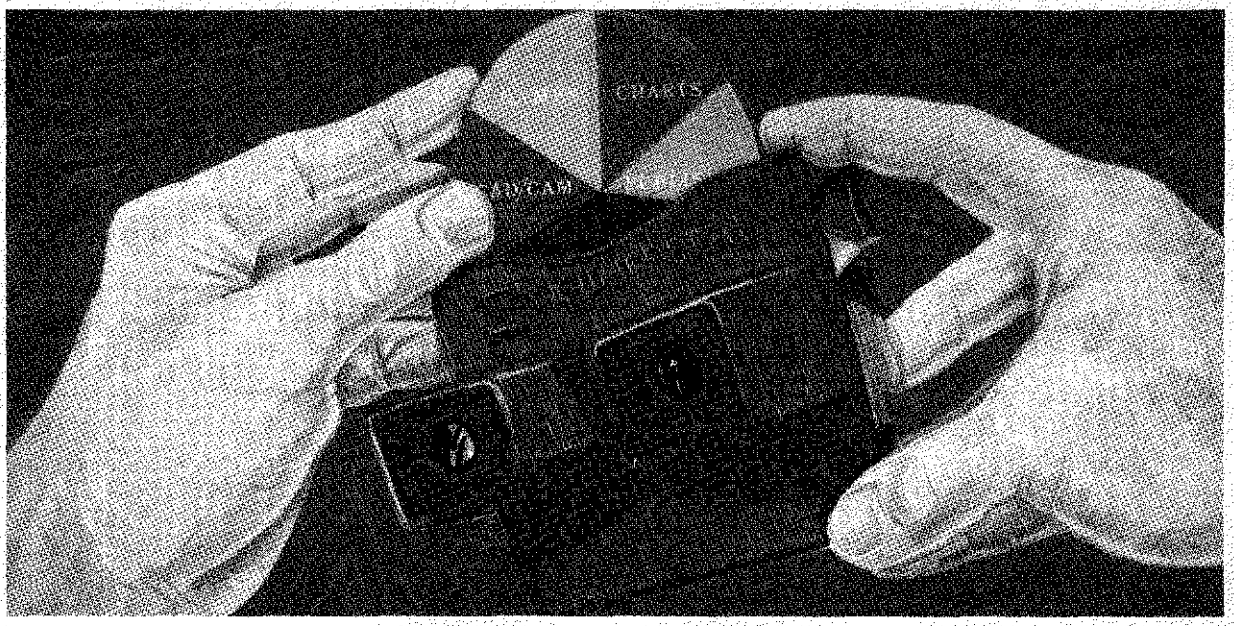
Custom Primitives

Since the 34010 is a fully functional processor, you can write code to make it perform special functions that TIGA does not provide, such as 3-D primitives, imaging functions, and even board-resident fractal generators.

Writing your own 34010 code requires the TMS340 SDK, and these don't come cheap (see the sidebar). Once you have them, though, writing custom routines is very straightforward.

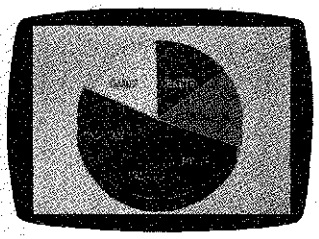
One approach is to write custom 34010 routines that use only existing TIGA primitives. This is a safe approach in that you aren't directly mucking with the 34010 graphics registers and display, and so you don't have to worry about supported board-dependent features, such as silicon for drawing acceleration. Another benefit of this clean approach is that you are able to develop and debug your code entirely on the PC side (providing you use C), since the TIGA calls for the PC- and board-based software are identical, thanks to the macros in the TIGA include files.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



We've Found a Better Way to Add 3-D To our C Graphics Package

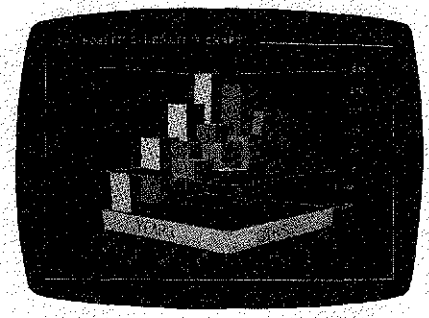
We've added some very powerful 3-D functions to our best-selling graphics package. These are true multi-view, multi-perspective capabilities, not a bunch of warmed-over 2-D stuff. This latest addition enhances Essential Graphics' position as "the" professional programmer's choice.



Updates to other functions are also impressive. Essential Graphics now supports more boards than ever, including Extended VGA 800x600 and 1024x768 modes and the IBM 8514.

All the functions write directly to the board, making them very efficient. The package is loaded with things no compiler library includes: PC Paintbrush PCX and PCC, spline and bezier curves, light pen/digitizer routines, multiple graphics page access, and hundreds more.

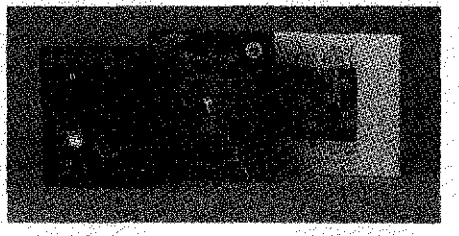
South Mountain believes that any company supplying sophisticated



True multi-view, multi-perspective charting capabilities.

code has an obligation to support that code. Call and ask for tech support before you buy our products and talk to the people who write and support the functions.

Some Satisfied Users



Dan Barstow, Accu-Weather: *"I couldn't have done it without Essential Graphics."*

Jeff Cooper, dBase Tools for C: *"Has everything a serious graphics programmer needs."*

Dale Young, TRACON: *"Essential Graphics is the cornerstone of TRACON."*

PC Magazine, 6/16/89: *"TRACON's graphics are superb."*

Features

- ◆ Over 300 functions
- ◆ Fastest functions available
- ◆ 3-D bar, pie, pyramid, etc.
- ◆ Charting capabilities
- ◆ Supports 8514, VGA, EGA, Herc, CGA and more
- ◆ World coordinates and viewports
- ◆ Vector and bit-mapped fonts
- ◆ Font rotation and scaling
- ◆ Hard copy to dot matrix printers, HP lasers and plotters
- ◆ PC Paintbrush support
- ◆ Block moves and animation
- ◆ Rubberbanding
- ◆ Line styles and widths
- ◆ Scanners ◆ Mouse support
- ◆ Pattern fills ◆ Multiple paging
- ◆ Digitizers and light pens
- ◆ Compilers Microsoft, Turbo, Lattice

Other Products

C-Utility, Essential Communications, Break0ut II
Essential B-Tree,
ScreenStar, /*resident_C*/

Price $299 - Source $299

No Royalties, 30-Day Money-Back Guarantee. Call for a free demo disk.

1-800-451-6174

SOUTH MOUNTAIN SOFTWARE™

76 S. Orange Ave., Suite 3
South Orange, N.J., 07079
NJ 201-762-6965 Fax. 201-762-0118

Essential Communications, Essential Graphics, C-Utility Library, ScreenStar, Essential B-Tree, /* resident_C*/ are trademarks of South Mountain Software, Inc., Viewmaster is a trademark of GAF Inc.

PRODUCED 2004 BY INTORG
ELECTRONIC REPRODUCTION PROHIBITED

install_alm	Install absolute load module	Core
install_font	Install font into font table	Extended
install_primitives	Install extended drawing primitives	Core
install_primitives	Install extended drawing primitives	Core
install_rlm	Install relocatable load module	Core
install_usererror	Install user error handler	Core
lmo	Return left-most-one bit number	Core
loadcoff	Load a COFF program	Core
page_busy	Return status of page flipping	Core
page_flip	Set display and drawing pages	Core
patnfill_convex	Draw patterned convex polygon	Extended
patnfill_convex	Pattern fill convex	Extended
patnfill_oval	Draw patterned ellipse	Extended
patnfill_piearc	Draw patterned pie slice	Extended
patnfill_polygon	Draw patterned polygon	Extended
patnfill_polygon	Pattern fill polygon	Extended
patnfill_polygon	Pattern fill polygon	Extended
patnfill_rect	Draw patterned rectangle	Extended
patnframe_oval	Draw patterned oval border	Extended
patnframe_rect	Draw patterned rectangular border	Extended
patnpen_line	Draw line with pattern and pen	Extended
patnpen_ovalarc	Draw oval arc with pattern and pen	Extended
patnpen_piearc	Draw pie slice with pattern and pen	Extended
patnpen_point	Draw pixel with pattern and pen	Extended
patnpen_polyline	Draw lines with pattern and pen	Extended
patnpen_polyline	Pattern pen polyline	Extended
peek_breg	Read from a B-file register	Core
pen_line	Draw line with pen	Extended
pen_ovalarc	Draw an oval arc with pen	Extended
pen_piearc	Draw pie slice with pen	Extended
pen_point	Draw point with pen	Extended
pen_polyline	Draw lines with pen	Extended
pen_polyline	Pen polyline	Extended
poke_breg	Write to a B-file register	Core
rmo	Return right-most-one bit number	Core
seed_fill	Fill region with color	Extended
seed_patnfill	Fill region with pattern	Extended
select_font	Select an installed font for use	Extended
set_bcolor	Set background color	Core
set_clip_rect	Set clipping rectangle	Core
set_colors	Sets foreground and background colors	Core
set_config	Set graphics config	Core
set_curs_shape	Set cursor shape	Core
set_curs_state	Make cursor visible/invisible	Core
set_curs_xy	Set current cursor position	Core
set_draw_origin	Set drawing origin	Extended
set_dstbm	Set destination bitmap	Extended
set_fcolor	Sets foreground color	Core
set_interrupt	Set an interrupt handler	Core
set_palet	Set an entire palette	Core
set_palet_entry	Set a palette entry	Core
set_patn	Sets current pattern description	Extended
set_pensize	Sets current pensize	Extended
set_pmask	Sets color plane mask	Core
set_ppop	Sets pixel processing operation	Core
set_srcbm	Set source bitmap	Extended
set_textattr	Set text rendering attributes	Extended
set_timeout	Set timeout timing value	Core
set_transp	Set transparency mode	Core
set_vector	Set contents of GSP trap vector	Core
set_videomode	Set emulation mode	Core
set_windowing	Sets windowing mode	Core
set_wksp	Set a temporary workspace	Core
styled_line	Draw styled line	Extended
swap_bm	Swap source and destination bitmaps	Extended
synchronize	Make host wait for GSP to idle	Core
text_out	Render an ASCII string	Core
text_width	Return the width of an ASCII string	Extended
transp_off	Disables pixel transparency	Core
transp_on	Enables pixel transparency	Core
wait_scan	Wait for a designated scan#line	Core
zoom_rect	Zoom source rectangle	Extended

A second approach to writing the 34010-resident code is to implement most, if not all, of the primitives yourself, even to the point of duplicating existing TIGA capabilities. This will almost always produce the tightest and fastest code (if you know what you're doing), but it may sacrifice compatibility on some boards, such as new ones based on the 34010's successor, the 34020.

Probably the best approach, though, is a combination of the two: Write a substantial portion of the code yourself, and rely on TIGA to handle board-specific features, such as color palettes. Your 34010 code is really just a custom extension of the GM, and TIGA provides the capability to dynamically load code with GM references in it and resolve the references; you can call your 34010 code and communicate with it from the PC via the standard methods provided by the AI.

When you develop your own custom routines, you create an RLM file. "RLM" stands for Relocatable Load Module, a file type that allows TIGA to place the code anywhere it wants in board memory. (Currently, in some cases, you may need to convert your RLM to an ALM [Absolute Load Module], which is memory-position sensitive, but TI is about to release TIGA 2.0, which will do away with ALMs entirely.)

Back to the Design Goals

In this brief overview of TIGA, I've tried to touch on all the areas necessary to understanding how TIGA functions. I have still left much uncovered, but you can call the 800 number listed in the sidebar to get a TIGA manual, which will help explain all these areas in much greater detail.

I think the TI engineers who designed TIGA pretty much met their goals, but with minor exceptions:

TIGA is relatively simple to port, but the porting kit is a bit on the expensive side. This might cause problems for small graphics-board companies that don't have much cash, but since hardware tends to produce greater revenue than software, it could balance out.

TIGA is easy to use—for programmers. For end-users who don't know about environment variables, some aspects of TIGA will appear to be voodoo. Unfortunately, due to the way DOS works, there isn't much that can be done about this. Also, the price of the SDK is a tad high for many small developers. (If you are serious about developing TIGA software and you have a good idea related to the development but are concerned about the cost of the development tools, call the TI Graphics Hotline and talk to them about it. Texas Instruments has given me the impression that they don't want to turn away any potential TIGA developers.)

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MORE 34010 Video Co-Processors from PC Tech

- Mono 736 x 1024 - 2048 x 1536
- Color 1024 x 768 (1280 x 1024 soon!)
- 512K - 4M DRAM
- Optional CCITT & DGIS support
- **AGE XoftWare X Window Server certified board**
- C Compiler, Assembler, Linker, Debugger, DGIS Development Kit & TIGA Toolkit available
- **TIGA** included with all boards
- Custom designs available
- **PC Tech understands the 34010**

For further information, please contact:

PC Tech
907 N. 6th St.
Lake City, MN 55041
(612) 345-4555 (Voice)
(612) 345-5514 (FAX)

PC Tech is a trademark of PC Tech, Inc.

PC, XT, AT, DGIS, XoftWare, and DGIS are trademarks or registered trademarks of their respective companies.

Number 208 on the Reader Service Card

META3D

New software!

Add 3D power to MetaWINDOW library

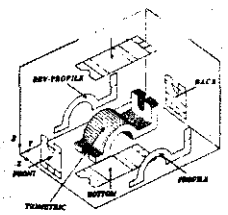
Develop powerful interactive 3D graphic application programs in C or FORTRAN faster and easier, display 2D/3D spreadsheet and experimental data using META3D, CGL3D and MetaWINDOW.

CGL3D — *A Complete 3D Color Graphic Library for your PC*

CGL3D/META3D

- *True 3D perspective*
- *2D/3D Views, Rotation*
- *2D/3D Conversion*
- *Multiple Viewport*
- *World Coordinate, no limit*
- *Automatic Line clipping*
- *3D surface data display*
- *CGA,EGA,VGA, and more*
- *User manuals and sample source codes*
- *2D/3D Point, Line, Text, Polygon, Fill and surface*
- *PC MOUSE Interface*
- *Libraries are written in FORTRAN 77 and compiled with Microsoft FORTRAN V4.*

META3D + MetaWINDOW



Display different views

Similar in most functions to Tektronix PLOT-10(TCS) mainframe graphic plus additional true 3D functions

To order,META3D + CGL3D, Please send $199 plus $3 for shipping to:

F.G.SOFT
Tel 213-542-5461
P.O. BOX 2152,
Hawthorne,CA 90251-2152

Number 307 on the Reader Service Card

ProGraphx

ProGraphx Toolbox™

Over **50** powerful hi-speed graphics routines for EGA/VGA graphics adapters, including proprietary icon routines to display color images.

ProGraphx Image Editor, a complete color paint package to design color icon images. Edit and view up to 50 icons at once. Simply paint an icon, then use the ProGraphx Toolbox™ routines to display it with amazing speed. Over **500** sample icons for use in your own programs.

Operation:Airlift, a full screen scrolling arcade style game with source code to demonstrate the power and speed of the ProGraphx Toolbox™.

All routines written in assembly for fastest possible performance.

Available for Turbo Pascal, or C.

ProGraphx Menus™

Menus and icons without the work!

Mouse driven pull-down menu system. Use a simple text file format to define menu structure along with event codes, then compile and use in application. Only 4 lines of code to add. Event driven menus can be changed and rearranged external to program without recompiling application. ***ProGraphx Menus***™ uses the fast routines in ***ProGraphx Toolbox***™ to create the fastest response possible. **Powerful, yet easy to implement!**

Also, function-key keyboard menus. Implemented in both text and graphics mode for conformity of applications. Simple to use.

Complete source code for menu-based microcoded processor simulator, along with memory and microcode editor. Provides examples of all windowing routines, in both text and graphics mode.

Available for Turbo Pascal, or C.

ProGraphx Runtime/Source

ProGraphx ToolBox™	$199
Toolbox with Source	399
ProGraphx Menus™	99
Menus with Source	199
ProGraphx Demo Disk	9.95

*Specify Compiler and Version.
**Illinois Residents add 8% sales tax.

ProGraphx
Speed, Simplicity, and Power for Graphics Programmers

Send Check or Money Order to:
Stanwood Associates, Inc.
303 East Ohio Street
Chicago, IL 60611

With VISA/MC Call
1-(800) 752-7982
(312) 828-9734
FAX: (312) 828-9742

Number 330 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

TIGA is extensible, and with TIGA 2.0, a number of headaches (involving primarily ALMs) will be gone. This allows you to take full advantage of the 340X0 chip on the board without having to sacrifice portability—if you implement your software correctly.

TIGA can be a high-performance interface. The reason I say "can be" is because you might need to write your own board-resident code in order to squeeze the best possible performance out of a 340X0-based board. What this really depends on is your application. If your application spends most of its time processing information and is not graphics-speed bound, then you probably don't need to develop custom board-based routines. TIGA by itself, without custom code, performs quite well.

You might ask, then, "Why bother using TIGA?" Well, TIGA provides a base upon which to build your custom code. Without TIGA, you would have to handle each board as a separate entity, since virtually no boards out there handle functions like palettes or display initialization in a compatible fashion. But by using TIGA, you can mix and match performance with portability by customizing only your time-critical code and using generic TIGA primitives for the rest. TIGA compatibility makes your custom code portable as long as you avoid anything that is board specific; that is, as long as the code limits itself to the innate capabilities of the 34010, you're all set.

And Now . . .

The current version of TIGA (as I write this) is 1.1. According to TI, the following improvements will occur in 2.0, due for release this spring:

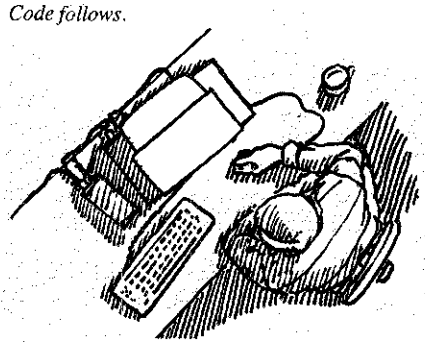
- TIGA 2.0 will be bimodal, meaning that it will run in both real and protected modes. The protected-mode support is designed to work with PharLap's and Rational System's DOS extenders.
- It will be fully 34020 compatible. Currently, there are some minor compatibility problems with the 34020, and 34020-specific instructions are not supported.
- The initialization information for TIGA will be in a separate DOS file to allow for quick configuration changes. Currently, that information is hard coded.
- The board-based memory manager will become handle based, allowing for better use and control of the on-board memory.
- ALMs will not be needed.
- Support for GSP to PC calls may be provided.
- IEEE-compatible floating-point data will be transferable and interpretable.
- Unlike version 1.1, TIGA 2.0 will not require you to allocate a large workspace out of board memory to draw filled polygons.

In Conclusion

Texas Instruments has put a lot of effort, time, and money into the development, support, and promotion of TIGA. This work will ultimately benefit us as users: Because our application packages will have TIGA drivers, we won't have to complain about how our XYZ Pixel-Masher Graphics Controller is not supported; and TIGA will benefit us as programmers, too, allowing us to write one driver to support a majority of the high-end graphics boards in the market. PJ

Jake Richter is the President of Panacea Inc., a PC-graphics consulting company based in Londonderry, New Hampshire, which specializes in custom software for graphics processors and graphics boards. Jake also writes graphics columns for Personal Workstation Magazine *and is anxiously awaiting the release of his book* Programming the 8514/A *(M&T Publishing), which should be appearing in bookstores soon.*

Code follows.



The Heap Expander™ version 2.05

Now your programs can have virtually unlimited heap space using expanded memory, extended memory, disk space, or any combination of the three. And it's all transparent. The Heap Expander's initialization code checks the system's resources and uses whatever is available.

only $79.95*

- Uses LIM-standard expanded memory if present.
- Uses AT-style extended memory if present.
- Swaps data to disk as needed.

Libraries and Source Code for:

- Turbo C
- Microsoft C 4.0 and 5.0

Requires an IBM PC, XT, AT, or close compatible with MS-DOS or PC-DOS version 2.0 or above

MC/VISA/COD call
1-800-248-1045 x 100 (US)
1-800-952-5560 x 100 (Idaho)

*CA residents add 6 1/2% sales tax. Foreign customers, please add $4.00 for shipping and handling.

The Tool Makers
P. O. Box 2151
Santa Cruz, CA 95063
(408) 458-0690

Number 166 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

All code in this issue is available through our bulletin board services or directly from *PJ* on disk. See pages 8, 88, 22, and 93.

Code from "Put a TIGA in Your PC," by Jake Richter

1

LISTING 1

```
/* SAMPLE.C

   Sample TIGA program written by Jake Richter, Panacea Inc.
   Use Microsoft C v5.1 to compile:

      CL SAMPLE.C \tiga\ai\ai.lib
*/
#include <typedefs.h>                    /* These are the TIGA type defs */
#include <tiga.h>                        /* TIGA Core primitive defs     */
#include <extend.h>                      /* TIGA Extended primitive defs */

/*
   Useful definitions using TIGA configuration information.
*/
#define  XSIZE     config.mode.disp_hres
#define  YSIZE     config.mode.disp_vres
#define  CHARHGT   fontInfo.charhigh
#define  CHARWID   fontInfo.charwide

static   CONFIG    config;               /* TIGA Configuration Info      */
static   FONTINFO  fontInfo;             /* Font information             */
static   CHAR      testString[] = "Read Programmer's Journal!";

main(argc, argv)
INT   argc;
CHAR  **argv;
{
   INT     i, tmpWidth, tmpHeight;

   /* First we need to switch into TIGA mode. This installs the GM on
      the board if it isn't already present and initializes the board
      into the default resolution (set by the TIGAMODE utility).

      The "INIT" specifies that all GM globals and memory managment
      should be initialized, and the "CLR_SCREEN" tells TIGA to clear
      the screen to color 0. The function returns a nonzero value
      to indicate an error has occurred.
   */
   if (!set_videomode(TIGA, INIT | CLR_SCREEN)) {
      printf("TIGA not installed!\n");
      exit(1);
      }

   /* Next, we need to install the extended primitives, since line
      and rectangle drawing are not part of the core. If this function
      call returns a negative number, an error occurred.
   */
   if (install_primitives() < 0) {
      printf("Can't locate TIGA primitives!\n");
      exit(2);
      }

   /* Now we need to find out something about the board, primarily what
      resolution it is running. The resolution values have already been
      defined by the XSIZE and YSIZE macros listed earlier. While
      normally you should check to see how many colors (or bits per
```

2

```
      pixel) you have, it really isn't necessary for this example.

      Let's also get font information while we're at it. Normally we'd
      check if a font were available by getting the return value from
      get_fontinfo, but for now, let's just assume that one is available.
   */
   get_config(&config);
   get_fontinfo(0, &fontInfo);

   /* Let's do some drawing! First, let's draw a bunch of center-filled
      rectangles, going from largest to smallest, in different colors.

      The fill-rectangle command requires the rectangle width, height,
      and position.
   */

   for (i = 1; i < 6; i++) {
      set_fcolor(i);
      tmpWidth = XSIZE / i;
      tmpHeight = YSIZE / i;
      fill_rect(tmpWidth, tmpHeight,
               (XSIZE - tmpWidth) / 2, (YSIZE - tmpHeight) / 2);
      }

   /* Now let's draw an "X" that crosses the entire screen. The line-
      drawing command requires actual start and end points, and since
      the coordinate system starts at 0,0, we need to decrement the
      extents to get the maximum pixel position, especially if we are
      not sure how the clipping rectangle is set.

      By setting the color to -1, you generally ensure yourself that
      you will be drawing in bright white, since most color palettes are
      configured with white as the highest numeric color available.
   */
   set_fcolor(-1);
   draw_line(0, 0, XSIZE - 1, YSIZE - 1);
   draw_line(XSIZE - 1, 0, 0, YSIZE - 1);

   /*
      For the coup de grace, write some text, centered on screen.
   */
   text_out((XSIZE - CHARWID * strlen(testString)) / 2,
            (YSIZE - CHARHGT) / 2,
            testString);

   /*
      Wait until our masterpiece has been viewed by waiting for a
      keystroke, and then return the TIGA board back into its previous
      display state. This is extremely important, because most TIGA
      boards will either have a VGA pass-through or some type of
      IBM standard emulation mode to switch back to. In a one-monitor
      configuration, you can leave the user in trouble if you don't
      tell TIGA to go back to whatever emulation mode it was in prior
      to running your program.
   */
   getch();
   set_videomode(PREVIOUS, NO_INIT);

   return(0);                            /* No errors.                   */
}
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Reading the Unreadable EGA Palette

by
Christopher A. Howard

When it was introduced, the Enhanced Graphics Adapter (EGA) was a major improvement over the old Color Graphics Adapter (CGA). It offered more video modes, higher resolutions, more colors, and a host of programming features. Unfortunately, many of the EGA registers are write only; because they are unreadable, it is impossible to tell the exact status of the EGA at any given moment. This is normally very important for many terminate-and-stay-resident (TSR) applications, such as mouse drivers and screen-capture programs, and for any TSR designed to "pop up" in the graphics modes. One of the most aggravating implications of this is the inability to read the EGA palette.

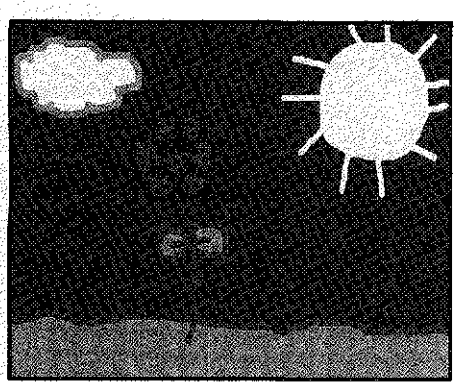
The palette is necessary for recreating the exact graphic image currently being displayed—saving the display memory is not enough. You can think of the available palette as the deluxe crayon set—you know, the one you never could get your mother to buy, with colors ranging from burnt sienna to Serengeti green. The display palette is a subset of the larger available palette (the standard crayon box you always had). At any given moment, you can have only the colors that will fit in your small box. For the EGA, this is usually expressed as 16 colors out of a palette of 64. If you think of display memory as your coloring-book drawing, you can choose any set of crayons you want to color it. Of course, some colors look more appropriate than others—a green sky with blue grass suggests you have picked the wrong crayon set (palette).

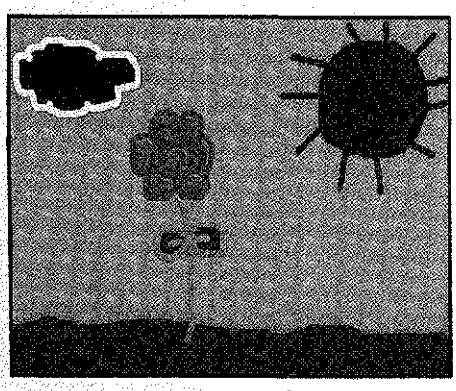
Now that we know why we need the palette, how do we get it? I know

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

of three methods. The first, trivial one works when the currently active program is under your control. If you displayed the picture and set the palette, you obviously know what it is and can save it without actually having to read the palette registers.

A TSR program does not have this liberty, and so another option is to trap the BIOS INT 10H vector and filter out the **set palette** function call. Although this will work, it slows down every INT 10H function; it is also hard to implement and prone to error.





The final method involves utilizing a little-known EGA BIOS table. Located in the EGA (or VGA) ROM BIOS, this table consists of seven double-word pointers to other important video BIOS locations, such as the Video Parameter Table pointed to by the first vector. The second double-word vector—the one of interest to us—points to a structure called the Dynamic Save Area. This area is a 256-byte buffer that the BIOS uses for storing pertinent information. The first 17 bytes consist of the last palette update plus the overscan value. The save area is updated every time the BIOS is called upon to modify the display palette. Initially, the save-area vector is filled in as 0000:0000H, which tells the BIOS to ignore it. We need to change this vector, but how do we update a table located in read-only memory?

Conveniently, there is a location in lower system memory that is simply referred to as the SAVE_PTR. It is a double-word (far) pointer to the Dynamic Save Area, and it will initially be set to some location in the ROM BIOS, in segment C000H. The SAVE_PTR itself is located at 0000:04A8H. The trick is:

1. Copy the 7-DWORD (14-byte) table out of BIOS ROM and into system RAM.
2. Update the save-area table entry so that it points to a 256-byte save buffer.
3. Update the SAVE_PTR value in lower memory so that it points to the new RAM table.

Simple enough?

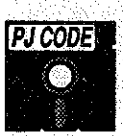
The assembly language code in Listing 1 is adapted from the routines we include in The PCX Programmer's Toolkit. It consists of a pair of functions, **pcxBeginPaletteGrab**, which installs the new table and work area, and **pcxEndPaletteGrab**, which removes it. It is necessary to remove the RAM table because the table is, in essence, part of an interrupt-service routine. If a program terminates without resetting the table, the next program will get partially overwritten the next time the work area is updated; this, of course, has some nasty consequences.

This method has worked reliably in our own TSRs and our users' programs for several years. Once installed, the current palette is always available without slowing down BIOS calls or the system. Remember that the palette has to be set at least once before the data in the buffer is valid. A good method for verifying this is to initialize the buffer with some known value, such as FFH. The only time these routines will not work is when a program bypasses the BIOS and writes directly to the palette registers. Fortunately, this is rare. If you deal with graphic images and palettes at all, these functions will be a useful addition to your programming library.

Chris Howard is the president of Genus Microprogramming, Inc. Genus has a number of products supporting graphics modes up through Super-VGA, including the PCX Programmer's Toolkit, PCX Effects, and PCX Text. He can be reached at Genus Microprogramming, 11315 Meadow Lake, Houston, TX 77077, or on CompuServe, user number 75100,17.

Code follows.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



All code in this issue is available through our bulletin board services or directly from *PJ* on disk. See pages 8, 88, 22, and 93.

Code from "Reading the Unreadable . . . ," by Chris Howard

1

LISTING 1

```
Palette Grab Procedures

; pcxPG.ASM                                                          ;
; Copyright (c) Genus Microprogramming, Inc. 1988-89  All Rights Reserved.   ;

;
; This file contains procedures for manipulating the EGA Work area.  This is used for
; retrieving the last BIOS set-palette call.
;
; Note that these routines are written using the Pascal model and that all code and
; data uses far references. This is all taken care of in the C function declarations.
;
; Procedures: pcxBeginPaletteGrab
;             pcxEndPaletteGrab
;
;   Assembly: MASM pcxPG
;
; Microsoft ASM 5.x version.              Programmer: Chris Howard  4/21/88
;

; Definitions
EGAtable  equ       04A8H                   ;EGA work area table pointer
EGAtablen equ       14                      ;Length of table
pcxmanuf  equ       0AH                     ;ID byte
pcxSUCCESS          equ     0               ;Success
pcxERR_NOPALGRAB    equ     -1              ;No Palette Grab started

   .model  small
   .data

pcxbiostb dw        ?                       ;Old BIOS table
          dw        ?

pcxidtb   db        ?                       ;Table ID byte (so we know us)
pcxparmtb dd        ?                       ;Video Parameter Table pointer
pcxsavetb dd        ?                       ;Dynamic Save Area pointer
          dd        ?
          dd        ?
          dd        ?
          dd        ?
          dd        ?

   .code

          PUBLIC    pcxBeginPaletteGrab,pcxEndPaletteGrab;

;
; This procedure begins the palette grab by redirecting the EGA Work vector
; to the specified buffer.
;
; Calling: retcode = pcxBeginPaletteGrab(palptr)
;
;

;Define variable locations on the stack  (pascal model)
bpgpalseg equ       <[bp+ 6]>
bpgpalofs equ       <[bp+ 4]>
bpgparm   equ       4

pcxBeginPaletteGrab PROC FAR

          push      bp                      ;Set up frame and save regs
          mov       bp,sp

          push      ds
          push      es
          push      si
          push      di

          xor       ax,ax                   ;Video BIOS Data pointer
          mov       ds,ax

          mov       ax,@data                ;Pointer to our table
          mov       es,ax

          mov       bx,EGAtable             ;Point to existing table
          lds       si,ds:[bx]
          mov       di,OFFSET pcxparmtb     ;Offset of our table
          mov       cx,EGAtablen
          rep       movsw                   ;Copy it

          mov       ax,@data                ;Point to our data again
          mov       ds,ax

          mov       BYTE PTR pcxidtb,pcxmanuf ;ID our table, for "pcxEndPG"

          mov       ax,bpgpalofs            ;Point to save area
          mov       WORD PTR [pcxsavetb],ax
          mov       ax,bpgpalseg
          mov       WORD PTR [pcxsavetb+2],ax

          xor       ax,ax                   ;Point to BIOS table pointer
          mov       es,ax

          cli                               ;Disable interrupts
          mov       bx,EGAtable             ; and point BIOS to our table
          mov       ax,WORD PTR es:[bx]     ;Save pointer
          mov       pcxbiostb,ax
          mov       ax,WORD PTR es:[bx+2]
          mov       pcxbiostb[2],ax
          mov       WORD PTR es:[bx],OFFSET pcxparmtb  ;and set
          mov       ax,ds
          mov       WORD PTR es:[bx+2],ax
          sti                               ;Enable interrupts

          mov       ax,pcxSUCCESS           ;Return

pcxBPG_exit:


          pop       di
          pop       si
          pop       es
          pop       ds

          mov       sp,bp
          pop       bp
          ret       bpgparm

pcxBeginPaletteGrab ENDP
```

2

Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

EGA VGA SVGA Unix Xenix Graphics Tool Kit

C Source and Binary Versions: *both include C source to a full-featured GIF viewer and several other demonstration programs.*

No CGI or BIOS - Direct Video Card Access: *Fast!*

12 Video Modes: *including 800x600x16 and 360x480x256 on VGA systems, and 640x400x256 on Paradise SVGA cards.*

Multiple Active Graphics Virtual Terminals: *Compatible with X Windows and VPix.*

Primitives: *point, line, solid, bitblt, clipping, ellipse, region fill, more. Twelve alu operations in 16-color video modes. And programs can be written independent of video mode and adapter.*

Software Cursor for Bus Mouse. Fifty BDF Fonts.

Fully Documented and No Royalties.

BONUS: The best in Unix Public Domain graphics C source code with drivers for the Unix/Xenix Tool Kit, including:

A device-independent (Unix/Xenix, DOS, POSTSCRIPT, X11) 2-d and 3-d plot library featuring 3-d transformations (including fonts) and various types of curve fits.
Ray tracers with output display, manipulation and image format conversion utilities.

Operating System Version	Source	Binary
Interactive 386/ix (all versions)	$199	$99
Xenix/386 2.3	$199	$99
AT&T System V/386	$199	$99
others	CALL	CALL
GIF viewer binary (specify Unix version)		$39

Send for More Information and a Demo!

JDyx *Enterprises*
907 Tuxworth Circle, Decatur, GA 30033
(404) 320-7624 BBS: (404) 321-5020

Number **346** on the Reader Service Card

Tired of the limitations of MS-DOS ?
Thinking about switching to another OS ?
Well DON'T, now there is :

PRO-CLI v4.1

CLI is an intelligent front & back end processor that enhances the MS-DOS system kernel. CLI provides many features of advanced OS's, like VMS and UNIX, to MS-DOS without any re-training.

The core of the CLI kernel is based on recursive definitions and references to Alias, Logicals, Symbols, and Lexical-Functions from the command line, batch programs, and existing application programs (without any modifications).

Powerful command programs can be quickly written for a wide range of purposes such as:

- flexible control of code & data management
- code development & code support tools
- automated file/device maintanence utilities
- creation of generic command procedures
- installations & system re-configurations

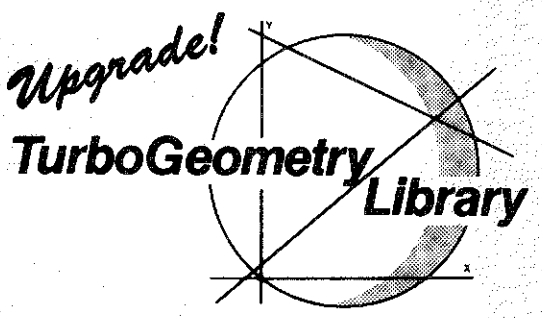
For the first time, designers, developers, and managers can coordinate software development and production efforts without the major limitations and restrictions in the MS-DOS environment.

* VMS is a trademark of D.E.C.
* UNIX is a trademark of AT&T
* MS-DOS is a trademark of MicroSoft

Alphax Systems Inc
1005 Sussex Blvd
Broomall, PA 19008

Price: $149
S/H: $ 6
215-543-2658

Number **345** on the Reader Service Card



A Library of 2D & 3D Geometric Routines

New 3.0 upgrade! Twice as many 2D and 3D geometric routines (Over 300). Now with: Surfacing • Solids • Spherical Geometry • 3D Clipping • Tangents. Also includes additional HideLine algorithms • Intersections • Volumes • Areas • Transforms • Perspectives • Decomposition • and more. Easy to use manual. 30 day guarantee. $199.95 with source. Domestic S&H Incl. Foreign S&H Extra. TurboGeometry Library 3.0 is available in Turbo Pascal, Microsoft C, & Turbo C. Needs MS/PC DOS 2.0+. IBM/PC or Compatibles. We accept VISA, MasterCard, PO, Chk. USA negotiable funds only.

Disk Software, Inc.
2116 E. Arapaho Rd.
Suite 487
Richardson, TX 75081
USA

214-423-7288
214-423-4465 FAX
800-635-7760

Number **229** on the Reader Service Card

Total control with *LMI FORTH*™

For Programming Professionals: an expanding family of compatible, high-performance compilers for microcomputers

For Development:
Interactive Forth-83 Interpreter/Compilers for MS-DOS, OS/2, and the 80386

- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 500 page manual written in plain English
- Support for graphics, floating point, native code generation

For Applications: Forth-83 Metacompiler

- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, 6303, 6809, 68HC11,34010, V25, RTX-2000
- No license fee or royalty for compiled applications

LMI *Laboratory Microsystems Incorporated*
Post Office Box 10430, Marina del Rey, CA 90295
Phone Credit Card Orders to: (213) 306-7412
FAX: (213) 301-0761

Number **63** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Howard code, continued

```
; This procedure ends the palette grab by redirecting the EGA Work vector
; back to its original location.
;
; Calling: retcode = pcxEndPaletteGrab()
;
; Note: Removal procedure is more complicated than just replacing the pointers
;       with our saved parms.  Using the below method allows us to write
;       programs like pcxGrab, where one copy of pcxGrab begins the palette
;       grab and another copy removes it (i.e., our data seg is not the same).
;
;

pcxEndPaletteGrab  PROC FAR
                   push  ds                  ;Save regs.
                   push  es
                   push  si
                   push  di

                   mov   ax,@data            ;Make sure DS = data
                   mov   ds,ax

                   xor   ax,ax               ;BIOS area
                   mov   es,ax

                   mov   bx,EGAtable         ;Point to the table
                   les   di,es:[bx]
                   cmp   BYTE PTR es:[di-1],pcxmanuf ;Look for our id byte . . .
                   je    pcxEPG_reset        ;If not us, error

                   mov   ax,pcxERR_NOPALGRAB ;Palette table is not us
                   jmp   SHORT pcxEPG_exit

pcxEPG_reset:      sub   di,5                ;Point to our saved parms
                   mov   si,es:[di+2]        ;Get the old pointer
                   mov   di,es:[di]

                   xor   ax,ax               ;Point to BIOS table pointer
                   mov   es,ax

                   cli                       ;Disable interrupts
                   mov   bx,EGAtable         ; and point BIOS to old table
                   mov   WORD PTR es:[bx],di
                   mov   WORD PTR es:[bx+2],si
                   sti                       ;Enable interrupts

                   mov   ax,pcxSUCCESS

pcxEPG_exit:       pop   di
                   pop   si
                   pop   es
                   pop   ds

                   ret

pcxEndPaletteGrab  ENDP

                   END
```


GRAPHICS!

Add lightning fast graphics to your programs quickly and easily through the popular PCX file format. Why reinvent the wheel? Make your programs immediately compatible with hundreds of packages from Aldus PageMaker to ZSoft's PC Paintbrush with these linkable graphic libraries.

"An exceptional product" - Programmer's Journal, Aug

NEW! Version 3.5 of the **PCX Programmer's Toolkit** gives you over 60 powerful functions to manipulate bitmapped graphics. Use Virtual screens, Super VGA modes, LIM 4.0 support, a 300 page manual, 9 utilities including screen capture and display, and the fastest routines on the market. **$195**

Need Special Effects, but caught in a GRASP?

Why create a demo when you can create the real thing? Don't be trapped in a slideshow editor or demo program when you can use **PCX Effects** for the PCX Toolkit and your favorite programming language. A Music Language and spectacular effects for exploding your graphics! **$99**

Blazing Graphics Text

With **PCX Text** you can display text with graphics as fast as it always should have been. Display characters, strings, fixed and proportional text, background transparency, and more. Includes a font editor, 85 fonts, and text utilities for blazing graphics bitmapped text. **$149**

All packages support 12 compilers for C, Pascal, Basic, Fortran, Assembly, and Clipper. All modes of the Hercules, CGA, EGA, VGA, and Super VGA adapters are supported, up through 800x600x256 (22 modes in all). Assembly Language source code is optionally available. Trademarks are property of their respective holders.

No Royalties! 30-day Money Back Guarantee.
VISA/MC/AMEX/COD/PO accepted.

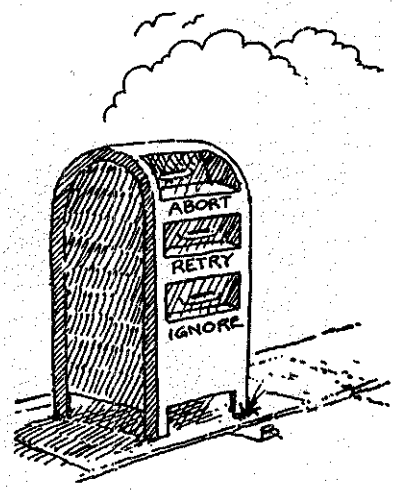
G·E·N·U·S MICROPROGRAMMING
11315 Meadow Lake • Houston, Texas 77077 • (713) 870-0737

1-800-227-0918

Number **261** on the Reader Service Card

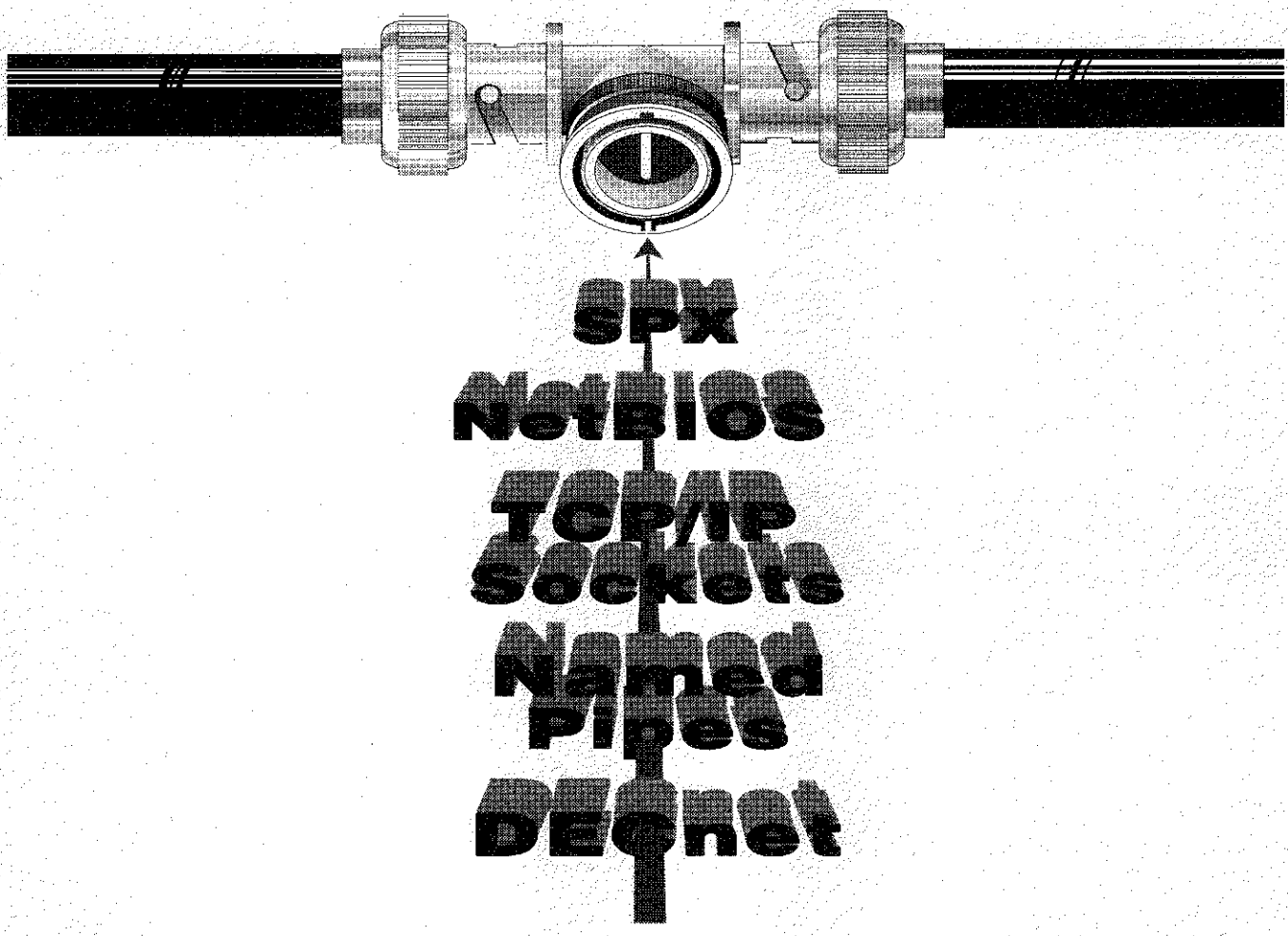
HOW'RE WE DOING?

Please let us know what you think of this issue—fill out the peer-review card facing page 88, and drop it in the mail. (The postage is on us.)



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Are networks creating obstacles for your applications?



Writing network communications software for your applications is difficult enough. Writing it for multiple networks or to connect applications running on different operating systems and hardware platforms is just too much. Too much time. Too much money. And too much energy searching for the networking software expertise you need to overcome your multi-vendor computing obstacles.

At Netwise, we believe your time and money are best spent at the applications level, building solutions for today's network computing environments — not struggling with writing low-level networking code.

The Netwise® RPC TOOL™ lets you breeze past the obstacles by providing access to a common software platform for the development of distributed applications. You write the application, then specify the network. The RPC TOOL automatically generates distributed processing software for you. All of it. Every time. And does it TODAY — for NetBIOS, NetWare, LAN Manager, DECnet, and TCP/IP networks. No waiting on promises. No vaporware. No excuses.

Banyan, Novell, Sun, 3Com and other major software and hardware suppliers have endorsed this technology as a standard for building distributed applications. Now it's your turn.

Call us for a FREE educational diskette. Or order a PC version of the RPC TOOL for a FREE thirty-day trial. Save your struggling for obstacles worthy of the effort.

Use the Netwise RPC TOOL once, and never write network communications code again.

Netwise is a registered trademark and RPC TOOL is a trademark of Netwise, Inc. Banyan is a registered trademark of Banyan Systems Incorporated. DECnet is a trademark of Digital Equipment Corporation. Novell is a registered trademark and NetWare is a trademark of Novell, Inc. Sun is a registered trademark of Sun Microsystems, Inc. 3Com is a registered trademark of 3Com Corporation.

NETWISE®

Call us. 800/733-7722
2477 55th Street,
Boulder, Colorado 80301
USA

See Us at UniForum, Booth #1347

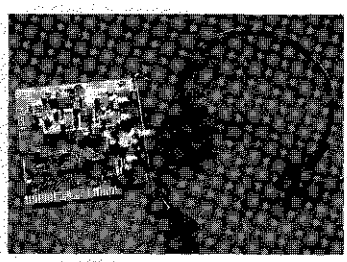
Call me, I'm interested: Circle 378 **Send Literature: Circle 278**

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

VOICE MASTER KEY
VOICE RECOGNITION SYSTEM
FOR PC/COMPATIBLES & TANDY 1000 SERIES
A FULL FEATURED VOICE I/O SYSTEM

GIVE A NEW DIMENSION TO PERSONAL COMPUTING. . .The amazing **Voice Master Key System** adds recognition to just about any program or application. Voice command up to 256 keyboard macros from within CAD, desktop publishing, word processing, spread sheet, or game programs. Fully TSR and occupies less than 64K. Instant response time and high recognition accuracy. Voice recognition tool-box utilities are included. **A genuine productivity enhancer!**

SPEECH RECORDING SOFTWARE. . .Digitally record your own speech, sound, or music to put into your own software programs. Software provides sampling rate variations, graphics-based editing, and data compression utilities. Create software sound files you can add to macros for voice recognition verification response. **a complete, superior speech and sound development tool.**



SOFTWARE CONVERSION CODES. . .The **Voice Master Key System** operates a growing list of third party talking software titles using synthesized phonetics (text - to - speech) or digitized PCM, ADPCM, and CVSDM encoded sound files. **Voice Master Key system does it all!**

EVERYTHING INCLUDED. . .**Voice Master Key System** consists of a plug-in card, durable lightweight microphone headset, software, and manual. Card fits any available slot. External ports consist of mic inputs and volume controlled output sockets. **High quality throughout, easy and fun to use.**

ONLY $149.95 COMPLETE

ONLY $89.95 FOR TANDY 1000 SL/TL MODELS SOFTWARE ONLY—REQUIRES TANDY BRAND ELECTRET MICROPHONE.

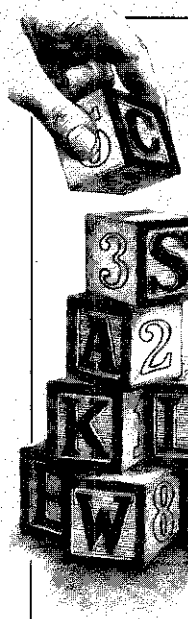
ORDER HOTLINE: (503) 342-1271 Monday-Friday, 8AM to 5 PM Pacific Time

Visa/MasterCard, company checks, money orders, CODs (with prior approval) accepted. Personal checks subject to 3 week shipping delay. Specify computer type and disk format (3½" or 5¼") when ordering. Add $5 shipping charge for delivery in USA and Canada. Foreign inquiries contact Covox for C & F quotes. 30 DAY MONEY BACK GUARANTEE IF NOT COMPLETELY SATISFIED. ONE YEAR WARRANTY ON HARDWARE.

CALL OR WRITE FOR **FREE** PRODUCT CATALOG

COVOX INC.
675 Conger Street, Eugene, OR 97402
TEL: 503-342-1271 • FAX: 503-342-1283

Number **277** on the Reader Service Card



Build with an Inside! advantage...

Thousands of C, BASIC, FORTRAN and Modula-2 programmers are already using Inside! to improve application performance. You have the best compiler and debugger, now add the performance analysis tool that helps create world class applications!

Inside! analyzes your program in real-time, timing each instance of a function, source line or arbitrary code fragment, all with microsecond accuracy and without source code modification. Utilize the following modes to know where to focus your optimization efforts for maximum reward.

- Function/procedure timing
- Call target timing
- Source Line timing
- Arbitrary event timing
- Non-executed functions or source lines
- MS-DOS function timing
- Overlay and assembly language support

Inside! is available for popular compilers . . .

Call or write today for a free brochure and the latest list of Inside! products.

To Place Orders
(800)537-5043
Product Support
(508)478-0499
Visa/Mastercard/C.O.D. Accepted

$125.00 EACH
Satisfaction guaranteed

Paradigm Systems
P.O. Box 152 Milford, Massachusetts 01757
Inside! is a trademark of Paradigm Systems.

Contact this advertiser directly.

HARVARD SOFTWORKS

NUMBER ONE IN FORTH INNOVATION

(513) 748-0390 P.O. Box 69, Springboro, OH 45066

MEET THAT DEADLINE ! ! !

- Combine the raw power of extensible languages with the convenience of carefully implemented functions!
- Execute faster than optimized C!
- Compile 40,000 lines per minute!
- Stay totally interactive, even while compiling!
- Program at any level of abstraction from machine code thru application specific language with equal ease and efficiency!
- Modify routines without recompiling!
- Use source code for 2500 functions!
- Use data structures, control structures, and interface protocols from any other language!
- Implement that borrowed feature, often more efficiently than in its source!
- Use an architecture that supports very small programs or full megabyte ones with a single version!
- Forget chaotic syntax requirements!
- Use subroutine libraries written for other languages! More efficiently!
- Outproduce experienced programmers stuck using conventional languages!

HS/FORTH with FOOPS - The only full multiple inheritance object oriented language under MSDOS!

WAKE UP ! ! !

Forth is no longer a language that tempts programmers with "great expectations", then frustrates them with the need to reinvent simple tools expected in any commercial language.

HS/FORTH Meets Your Needs!

We don't shortchange you with promises. We provide implemented functions to help you complete your application quickly.

Don't judge Forth by public domain products or ones from vendors primarily interested in consulting - they profit from not providing needed tools! Public domain versions are cheap - if your time is worthless. Useful only in learning Forth's basics, they fail at showing its true potential. Not to mention being s-l-o-w.

You can't add extensibility to fossilized compilers. You can easily add features from any other language to **HS/FORTH**. Try our full modifiable multiple inheritance **OOPs** extension or our **Fortran** translator & mathpak!

HS/FORTH has more features and libraries than we can possibly describe here. **Please ask for our brochure!**

HS/FORTH uses MSDOS or PCDOS. Each level includes features of lower ones. Level upgrades: $25. plus price difference. Sources in ordinary text files.

STUDENT LEVEL $145.
text & scaled/clipped graphic windows, ellipses, splines, arcs, fills, turtles; powerful parsing, formatting, file and device I/O; shells; interrupt handlers

PERSONAL LEVEL $245.
software floating point, trig, transcendental, quad & scaled integer, automatic optimizer for assembly like performance.

PROFESSIONAL LEVEL $395.
hardware floating point, turnkey, seal, Rosetta Stone Dynamic Linker, round robin & time slice multitaskers, dynamic string manager, file blocks, sector mapped blocks, assemblers

PRODUCTION LEVEL $495.
Metacompiler: DOS/ROM/direct/indirect C data structures & struct compiler, TurboWindow-C graphics library,

Options:
FOOPS+ with multiple inheritance **$ 75.**
286FORTH or **386FORTH** **$295.**
TurboWindow-C (Metagraphics) **$ 99.**
BTRIEVE for HS/FORTH (Novell) **$199.**
ROMULUS HS/FORTH from ROM **$ 95.**
FFORTRAN translator/mathpak **$ 75.**

Number **42** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Programming the 800x600x16 Super-VGA Mode

by

Christopher A. Howard

Many articles have been written about the lack of a standard in the Super-VGA (SVGA) modes (those modes beyond the IBM VGA's 320x200x256 color mode 13H). While this is essentially true, one mode does exist that I would dare call a standard. In fact, any program that supports the EGA 640x350x16 color mode (which was introduced five years ago) or the standard VGA 640x480x16 color mode can be easily modified to support the Super-VGA 800x600x16 mode.

This article will cover four Super-VGA chip sets—Tseng Labs', Video Seven's, ATI's, and Paradise/Western Digital's—illustrating how to set up the mode, address memory, and display pixels with each one.

Memory Planes

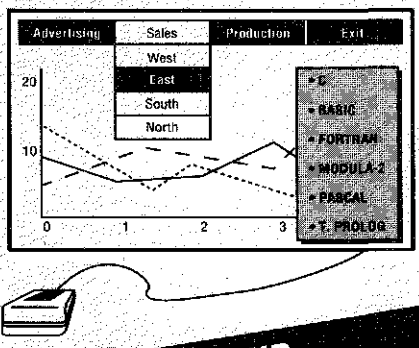
The problem with most Super-VGA modes is that each chip set handles graphics-memory configurations differently. Memory banks are selected, through different macros, by setting bits and selecting port registers. This mainly affects the 256-color modes, since they are so memory intensive. This bank switching is necessary for mapping the display memory into a 64K window, so that it may be accessed by the CPU. The 16-color modes handle the memory requirements by using four memory planes—one plane each for red, green, blue, and intensity. These are like memory banks, where each bank controls a color. However, the EGA/VGA/ SVGA adapters all handle planar memory in the exact same way.

In the EGA 640x350x16 color mode 10H, each plane takes up 28,000 bytes, for a graphics-memory total of 112K—which means a standard, 256K EGA can have up to two complete graphics pages. The VGA 640x480x16 color mode 12H occupies 38,400 bytes per plane—a 154K total. The Super-VGA 800x600x16 color mode sports 60,000

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

IF YOU PROGRAM, PLEASE TRY
HI-SCREEN XL™
WINDOWS - MENUS
DATA ENTRY - HELP

➤ **WYSIWYG** screen editor • draw • paint • copy • merge • undo • on-line ASCII table
➤ **WINDOWS** • overlapping • scrollable • for menus, data entry, help screens
➤ **MENUS** • pop-up • pull-down • scrolling • Lotus-style • "your-own-style"
➤ **HELP** • context-sensitive • at screen and field level • fully automated*
➤ **DATA VALIDATION** • type • range • format • color • justification • case
➤ **INPUT MANAGEMENT** • field-by-field and full-screen input • user-defined exit & function keys • arrays • hot keys



WITH MOUSE AND GRAPHICS SUPPORT*

➤ Use the **mouse** to select, scroll, validate, move cursor, click in dialog boxes, etc.
➤ Mix **graphics and text** screens in your programs. Capture any CGA, EGA or VGA graphic images and combine them with menus, windows, dialog boxes, etc.

Call toll-free: **1-800-338-2852**
in CA: 415-397-4666 • Fax: 415-397-4677

HI-SCREEN XL.....$149
HI-SCREEN XL *Professional Series*.....$325

30 day money-back guarantee • No royalties
* Professional Series version only

For C, BASIC, FORTRAN
COBOL, dBASE, PASCAL...

Softway, Inc.
500 Sutter #222, San Francisco, CA 94102

Number **110** on the Reader Service Card

Chipset	Adapter	Mode
Tseng	Orchid ProDesigner STB EM/16 Genoa Sigma X16	29H
ATI	VGA Wonder	54H
Paradise	Paradise Plus-16 Paradise Professional	58H
Video Seven	Video Seven VRAM VGA 1024i	62H (16H)

Table 1. 800x600x16 Mode Numbers

bytes per plane, for a total of 240K. Since each plane can still be accessed inside of one 64K window, no changes need to be made to the planar-memory format.

Initializing the Mode

The mode number is nonstandard and differs across the various graphics boards. Fortunately, this is the only nonstandard aspect of the 800x600x16 color mode, and it can be easily handled by a lookup table. A Super-VGA identification routine is not presented here, but one can either be hardcoded for your adapter or be set by the user via an installation procedure.

The mode is initialized like any other mode—through the BIOS SET MODE function. See Table 1 for a list of mode numbers for various VGA adapters. Only one adapter mentioned here, the Video Seven board, uses a different mode-setting scheme. However, I have found that its mode number of 62H is really a BIOS mode of 16H. If you use the BIOS and initialize the mode to 16H, you do not have to use Video Seven's alternate method.

Addressing Memory

The only other change involves pixel addressing. The standard 16-color modes have a display width of 640 pixels, or 80 bytes. For this Super-VGA mode, the width of the mode is 800 pixels, or 100 bytes. This affects the vertical-address calculation, because the *y*-coordinate must be multiplied by the number of bytes per line. The byte a particular pixel falls in can be found by a formula:

```
PixOfs = (y*bytesline)+(x/8)
```

Plotting the Pixel

After the byte has been found, the process of masking the pixel and writing it remains unchanged from any other EGA or VGA 16-color mode. In fact, it is still possible to use the same logical operations as those found in the normal 16-color modes (**replace**, **and**, **or**, **xor**). The routines presented here illustrate the use of logical operations, if the value PIXELOP is changed.

Once you can plot a pixel, you can do anything. Lines, circles, arcs, squares, ellipses—all can be derived from a pixel-plotting routine. And the logical operation can be used to make rubber-banded lines and boxes using the **xor** function.

Another Emerging Standard

The 800x600x16 color mode is the next emerging standard. With its higher resolution and one-to-one aspect ratio, it offers a smooth upgrade path. Best of all, it is easy to program and implement allowing most existing programs to be ported easily. Give it a try. PJ

Chris Howard is the president of Genus Microprogramming, Inc. Genus has a number of products supporting graphics modes up through Super-VGA, including the PCX Programmer's Toolkit, PCX Effects, and PCX Text. He can be reached at Genus Microprogramming, 11315 Meadow Lake, Houston, TX 77077, or on CompuServe, user number 75100,17.

Code follows.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

SOURCE

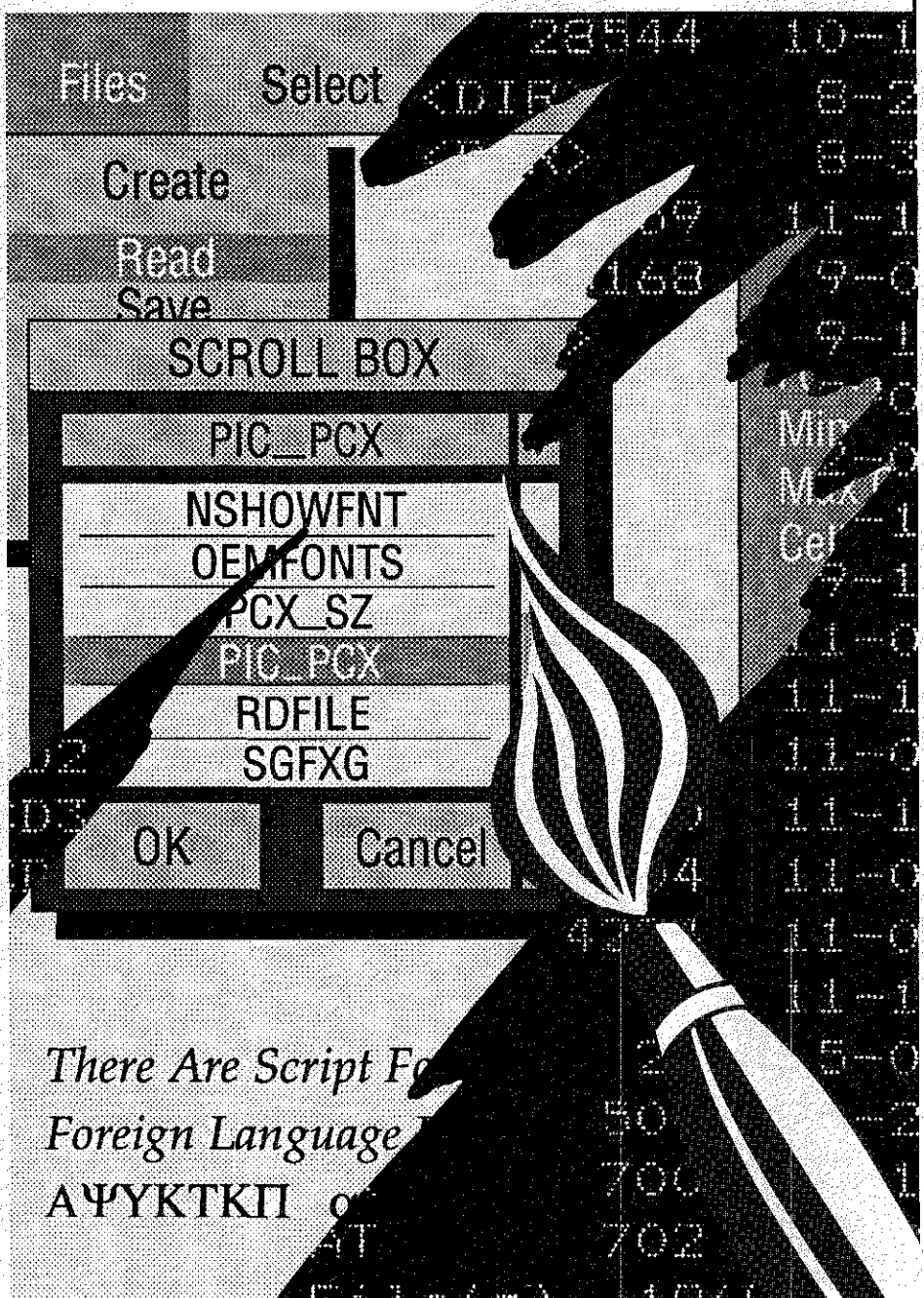
The GFX Libraries

NOW IN C, PASCAL & MODULA 2

The three GFX libraries have been translated from C to Pascal and Modula 2. No memory-hog TSRs to eat precious ram. Just small, fast linkable libraries. Each library stands alone. Use them together or with other graphics libraries. And of course, we provide all the source code.

Common features include:

Source Code: in your language.
No Royalties: for straight application programs.
Video Modes: Hercules, CGA, EGA, VGA (up to 800x16x16).
Supported Compilers: C — Microsoft, Borland, Lattice, Metaware, Zortech; **Pascal** — Borland (5.5); **Modula 2** — JPI and Stony Brook.



GFX Fonts & Menus Library

Now you can add a slick graphics user-interface to your programs: pull-down menus driven by mouse or hot-keys; dialog boxes; forms boxes; context-sensitive help; huge font selection (100+ fonts).
List Price: Source code - $150

GFX Font & Icon Editor

Not a library, but a fully blown editor. Make new fonts or modify fonts provided with GFX Fonts & Menus Library. Create new fonts or bitmapped icons. Use all 255 characters for fixed or proportional width fonts. Source code is only in C.
List Price: Source code - $100
Executable - $75

GFX Graphics Library

More power and speed than your compiler's library. Use virtual colors and auto-scaling to write code that adapts to any video mode at run-time. You also get rubber-banding and multiple viewports with automatic clipping and interger and floating-point scaling.
List Price: Source code - $150

GFX Screen Dump Library

Use this library to dump a graphics screen to a printer. We support most dot matrix, laser and color printers. Add new printers by filling in a table. Stretch the output to adjust for differences between video and printer resolutions.
List Price: Source code - $100
Executable - $75

■ **FREE BBS**

Extensive demos & info on all our products from a 24hr BBS.

BBS: (816) 478-0944
8 Data/1 Stop/No Parity

■ **SPECIAL OFFER**

Call us for special pricing for ordering direct from C Source.

Get our brochure on these and other programmers products.

C Source, Inc.
400 N.E. Point Dr.
Lee's Summit, MO 64064
TEL: (816) 478-1888
FAX: (816) 478-3133

Number 336 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

All code in this issue is available through our bulletin board services or directly from *PJ* on disk. See pages 8, 88, 22, and 93.

Code from "Programming the 800x600x16 . . . ," by Chris Howard

LISTING 1

```
; PutPix.ASM
; Copyright (c) Genus Microprogramming, Inc. 1988-89.  All Rights Reserved.
; Programmer: Chris Howard 12/22/89
;
; This file contains procedures for calculating the pixel offset in the 800x600x16
; color modes and plotting it.
;
; Procedures: PixelAddr
;             PutPix800
;
;   Assembly for C: MASM PutPix;
;   OR
;   Assembly for Turbo Pascal: TASM Putpix;

; Definitions
EGASEG              equ     0A000H              ;EGA/VGA/SuperVGA graphics seg
SUCCESS             equ     0                   ;Success

    .model tpascal    ;replace tpascal with SMALL for 'C' code

PixelREP  EQU       00000000B
PixelAND  EQU       00000001B
PixelOR   EQU       00000010B
PixelXOR  EQU       00000011B


;       .data                           ; Uncomment this line for 'C'
;       PixelOp   db        PixelREP    ; Uncomment this line for 'C'

        .code

; This function determines the address of a pixel in XVGA 800x600x16 color modes
; see pixtest.pas for mode to set with several XVGA adapters
;
; Calling: AX    = y-coordinate (0-599)
;          BX    = x-coordinate (0-799)
;
; Returns: ES:BX = pixel pointer
;          AH    = bitmask
;          CL    = bits to shift left
;

PixelAddr800        proc near

        mov       cl,bl                 ;CL = low-order byte of x
        push      dx                    ;preserve DX

        mov       dx,800/8              ;AX = y*bytesrow
        mul       dx

        pop       dx
        shr       bx,1
        shr       bx,1
        shr       bx,1                  ;BX = x/8
        add       bx,ax                 ;BX = y*bytesrow+x/8

        mov       ax,EGASEG             ;ES:BX = byte address of pixel
        mov       es,ax

        and       cl,7                  ;CL = x&7
        xor       cl,7                  ;CL = # bits to shift left
        mov       ah,1                  ;AH = unshifted bit mask

        ret

PixelAddr800        ENDP

;PutPix800(x,y,color)
;
; This function sets the value of a pixel in Super VGA (800x600x16). By
; setting PixelOp this routine will support REPLACE, AND, OR, and XOR
; logical operations also.

;Define variable locations on the stack  (turbo pascal model)
ppparm      equ       6
ppx         equ       <[bp+pppparm+4]>          ;Pixel coordinate
ppy         equ       <[bp+pppparm+2]>          ;"
ppcolor     equ       <[bp+pppparm]>            ;Color

        PutPix800 proc far
        local     pixelop:BYTE          ; Remove this line for 'C'

        public    PutPix800
;       push      bp                    ; Uncomment this line for 'C'
;       mov       bp,sp                 ; Uncomment this line for 'C'
        push      ds
        push      es

        mov       pixelop,PixelREP      ; Remove this line for 'C'
        mov       ax,ppy                ;Set up call to address routine
        mov       bx,ppx

        call      PixelAddr800          ;AH = bit mask
                                        ;ES:BX ->>>> buffer
                                        ;CL = # bits to shift left

        shl       ah,cl                 ;AH = bit mask in proper position
        mov       dx,3CEH               ;GC address register port
        mov       al,8                  ;AL = Bit Mask register number
        out       dx,ax

        mov       ax,205H               ;AL = Mode register number
                                        ;AH = Write Mode 2 (bits 0,1)
                                        ;     Read  Mode 0 (bit  3)
        out       dx,ax

        mov       ah,PixelOp            ;Get operation
        mov       cl,3                  ;Shift operation down into position
        shl       ah,cl                 ;AH = Read-Modify-Write bits
        mov       al,3                  ;AL = Data Rotate/Function Select reg
        out       dx,ax

        mov       al,es:[bx]            ;Latch one byte from each bit plane
        mov       al,ppcolor            ;Get the color
        mov       es:[bx],al            ;Update all bit planes

        mov       ax,0FF08H             ;Default Bit Mask
        out       dx,ax

        mov       ax,0005               ;Default Mode register
        out       dx,ax

        mov       ax,0003               ;Default Function Select
        out       dx,ax

gxPP_ok:
        mov       ax,SUCCESS            ;Return
        pop       es
        pop       ds
;       pop       bp                    ; Uncomment this line for 'C'

        ret

PutPix800  endp

        end
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

3

LISTING 2

```
{
  Pixtest.pas
  Copyright (c) Genus Microprogramming, Inc. 1988-1989  All rights Reserved
  Programmer: Chris Howard 12/22/89

  This is a simple test program for testing the PixelAddr and PutPixel functions.
  Turbo 5.0 was used to compile this program.  It draws a box bordering
  the 800x600 graphics screen.
}

{$A+,B-,D+,E+,F+,I+,L+,N-,O-,R-,S+,V+}
{$M 16384,0,16384}

program pixtest;

uses
  crt,
  dos;

{$L PUTPIX.OBJ}

procedure putpix800(x:word; y:word; color:word); external;

const

  SUCCESS=0;

var
  regs: Registers;
  i : word;
  ch : char;

begin
  { Display a header }
  writeln;
  writeln('Put Pixel Test Program');
  writeln('Press any key to begin,');
  writeln('Press any key when finished...');
  writeln;
  while not keypressed do;
  ch := readkey;

  { Now set the mode.  REMEMBER! This depends on your adapter! }
  { $29 800x600x16 Tseng     }
  { $58 800x600x16 Paradise  }
  { $62 800x600x16 Video7    }
  { $54 800x600x16 ATI       }

  regs.ax := $58;  { Currently set for Paradise VGA plus }
  intr($10,regs);

  for i:=0 to 599 do
  begin
    PutPix800(0,i,i div 16);     {Draw left side}
    PutPix800(799,i,i div 16);   {Draw right side}
  end;

  for i:=0 to 799 do
  begin
    PutPix800(i,0,14);   { Draw the top in yellow }
    PutPix800(i,599,15); { Draw the bottom in white }
  end;

  { Wait for a key }
  while not keypressed do;

  { Return to the text mode }
  regs.ax := 3;
  intr($10,regs);
end.
```


Save "Man-Years of Effort" with Turbo 5.5

Don't Start from Scratch with Object-Oriented Pascal

Object Professional is a huge library of over 200 object types and 2000 methods that will multiply your productivity. Window object types let you use overlapping and resizeable windows. The windows include ■ scrolling data entry screens ■ pick lists ■ menus ■ file selection ■ printed forms ■ help capability and more.

Build your programs using proven data object types like stacks, linked lists, virtual arrays, and more. System-oriented routines provide swappable TSRs in only 6K of RAM, EMS management, and much more.

Object Professional includes clear, comprehensive documentation, on-line help, full source code, technical support, and hot demo programs. Pay NO royalties. You'll get up to speed fast with OOP!

"The range of objects is fantastic. Object Professional could literally save you man-years of effort."

Jeff Duntemann

Object Professional 1.0, only $150.

A Multi-User B-Tree Toolkit

Write powerful network compatible databases faster and easier using **B-Tree Filer 5.0**. You'll have the fastest, safest, most flexible databases – no rigid structure, no TSR hassles, no running out of files. And they're compatible with Novell, 3Com, MS-NET, and others.

You get ■ Fixed and variable length records ■ Two billion records per database ■ Up to 100 indexes per index file ■ Fail-safe mode with journaling ■ Units for sorting, browsing, reindexing, and network control.

B-Tree Filer includes full source code, documentation, technical support, and you pay NO royalties.

"B-Tree Filer... a well rounded, feature-rich approach to B-Tree databases."

Computer Language, 1/90

B-Tree Filer 5.0, only $125. (single user)
With network support, $175.

Call toll-free to order.
1-800-333-4160

Satisfaction guaranteed or your money back within 30 days. Add $5 per order for shipping in U.S. and Canada. Inquire about other shipping charges. OPro requires Turbo 5.5. *BTF requires Turbo 4.0, 5.0, 5.5, or QuickPascal.*

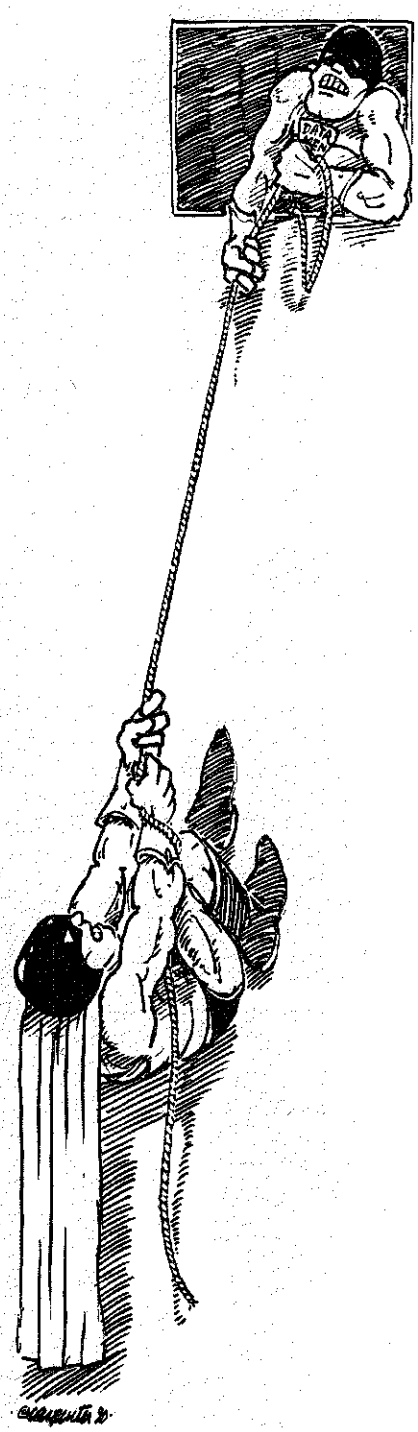
TURBO Power

8AM - 5PM PST Monday through Friday, USA & Canada.
For more information call (408) 438-8608. Fax: (408) 438-8610.
TurboPower Software PO Box 66747 Scotts Valley, CA 95067-0747

Number 25 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Communications Using Dynamic Data Exchange



by

William S. Hall

In the last issue of *Programmer's Journal* (Vol. 8.1, January/February 1990), I presented an overview of dynamic data exchange, or DDE, in Microsoft Windows. This time, I am providing a concrete realization of DDE from the realm of asynchronous serial communications. Consisting of two cooperating Windows programs, these are by far the most complex of any we have looked at in these pages. Seemingly straightforward at first glance, DDE can be surprisingly difficult to implement, and both the server and client applications presented here have required considerable attention to detail to function acceptably.

We begin our discussion with a brief review of DDE and of how its principles can be interpreted in communications. This leads us to a general discussion of how the server, COMSRV, and its client, COMCLI, work together to provide a simple terminal program in Windows. Finally, some of the implementation details from each program are exhibited.

One of the most interesting side issues in COMSRV is how the WM_DDE_EXECUTE message is handled. Sent by the client to set the communications port parameters, the command string is parsed using the Lex and Yacc programs supplied by Mortice Kern Systems (MKS, see the box at the end of this article). Designed basically to read and write stream I/O, MKS Lex and Yacc require some additional effort to function in Windows. Here we look at one way to use these powerful programs in an unusual setting.

All source and object code discussed in this article, including the Yacc and Lex .Y and .L files, are supplied on the *PJ* code disk for this issue. However, the .C and .H files produced by Lex and Yacc cannot be legally distributed. Hence, if you wish to modify the source accompanying this article, you may need the MKS Lex/Yacc system to do so. Also note that the makefiles have been written to use Microsoft's incremental linker, ILINK. On occasion, ILINK will not (for various reasons) automatically perform a full link; in these cases, you can use the accompanying batch file, FLINK.BAT, to link the program.

Together, COMSRV and COMCLI provide a reasonably good dumb-terminal program in Windows. It is reliable

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

and robust, and it works quite well against remote applications such as MCI Mail. And, because the server can handle as many clients as there are functioning communications ports, multiple simultaneous sessions are possible. COMSRV will even cooperate with Microsoft Excel in a simple way (see below). Nevertheless, neither the server nor the client contains a complete implementation of DDE. In particular, we have avoided the very complex problems of handling errors in reading and writing the communications ports and in tracking certain messages and their acknowledgments. The reader should be aware of these limitations in the programs presented.

Let us begin by reviewing some important facts about DDE. Please refer to the earlier article for additional details.

Some Elements of DDE

Windows has three mechanisms for exchanging data with other applications: clipboard transfers, shared memory, and message passing. DDE combines aspects of all three methods to provide a user-independent, interprocess communication protocol. Of course, the user is generally expected to establish the links manually, but once in place, each application shares the data automatically, either as it becomes available or upon specific request.

A three-level communications hierarchy expressed in terms of application, topic, and item is employed to achieve connectivity. A client requiring specific information can request it by identifying the program possessing the data (application), the logical data context (topic), and the specific component (item).

Transport of the data is accomplished via the Windows message system. Nine predefined messages are available to initiate and terminate a session, to execute a server command, to send data to or get data from the server on a one-time basis, and to create and destroy a "hot link" between the server and the client. The messages are combined in various ways to define complete transactions. A sequence of these transactions constitute a DDE session or conversation.

A typical DDE message has the following format:

```
PostMessage(hWndTarget, WM_DDE_XXX, hWndSender,
            MAKELONG(w1, w2));
```

Hence, each message contains but one 32-bit quantity, broken into two 16-bit parts, to carry the various data items that must be exchanged. Application, topic, and item are described by character strings but are encoded as 16-bit quantities called *global atoms*. The datum itself is defined by a handle to a globally allocated memory object, in which the first two words may contain flags and a registered clipboard-format value. Finally, the results of an exchange are described as bit fields of an unsigned short integer.

DDE in Serial Communications

Communications is a natural application of DDE. It is easy to imagine a single-server application that can deal with all aspects of the communications interface while handling several clients. At a minimum, such a server would have to be able to open and close the available ports, set parameters, and read and write both data bytes and control bits. The server could be extended to handle other items, such as file transfer, using any of several protocols.

Windows has three mechanisms for exchanging data with other applications: clipboard transfers, shared memory, and message passing.

In contrast, a client application, relieved of the complication of handling port details, can concentrate on processing and presenting the byte stream from the server according to various display protocols. It is easy to imagine, for example, a VT100 client and a Tektronix emulator talking to the same server, which itself is handling multiple hosts. The clients themselves may be main windows of different applications or children of a single application using the Windows Multiple Document Interface.

Having the server and the client as separate applications renders another advantage, namely, multiple threads of operations. Thus, operations that are modal to the client will not suspend the update stream from the server. For an illustration, compare the operations of Windows TERMINAL (*PJ* 7.6, see references) and COMCLI/COMSRV in the following way:

First, use TERMINAL to connect to a remote host and start a long typeout. Bring up the "About" box, and

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

A Short DDE Session Between COMSRV and COMCLI

```
Comcli[0]:WM_DDE_INITIATE 0x17f4->0xffff L=0xc02c H=0xc030 Comsrv COM1
Comcli[1]:WM_DDE_INITIATE 0x17f4<-0x17f4 L=0xc02c H=0xc030 Comsrv COM1
Comsrv[0]:WM_DDE_INITIATE 0x1508<-0x17f4 L=0xc02c H=0xc030 Comsrv COM1
Comsrv[1]:WM_DDE_ACK 0x1634->0x17f4 L=0xc02c H=0xc030 Comsrv COM1
Comcli[2]:WM_DDE_ACK 0x17f4<-0x1634 L=0xc02c H=0xc030 Comsrv COM1
Comcli[3]:WM_DDE_EXECUTE 0x17f4->0x1634 L=0 H=0x2ae
        [SetDCB("1200,8,0,0,,,,1,,,,,,1,1,,1,,,,17,19,32,32")]
Comsrv[2]:WM_DDE_EXECUTE 0x1634<-0x17f4 L=0 H=0x2ae
        [SetDCB("1200,8,0,0,,,,1,,,,,,1,1,,1,,,,17,19,32,32")]
Comsrv[3]:WM_DDE_ACK 0x1634->0x17f4 L=0x8000 H=0x2ae #32768 #686
Comcli[4]:WM_DDE_ACK 0x17f4<-0x1634 L=0x8000 H=0x2ae #32768 #686
Comcli[5]:WM_DDE_ADVISE 0x17f4->0x1634 L=0x2b6 H=0xc02c 1 0 1 1 Get
Comsrv[4]:WM_DDE_ADVISE 0x1634<-0x17f4 L=0x2b6 H=0xc02c 1 0 1 1 Get
Comsrv[5]:WM_DDE_ACK 0x1634->0x17f4 L=0x8000 H=0xc02c #32768 Get
Comcli[6]:WM_DDE_ACK 0x17f4<-0x1634 L=0x8000 H=0xc02c #32768 Get
Comcli[7]:WM_DDE_POKE 0x17f4->0x1634 L=0x2ba H=0xc02c 1 1 A Send
Comsrv[6]:WM_DDE_POKE 0x1634<-0x17f4 L=0x2ba H=0xc02c 1 1 A Send
Comsrv[7]:WM_DDE_ACK 0x1634->0x17f4 L=0x8000 H=0xc02c #32768 Send
Comcli[8]:WM_DDE_ACK 0x17f4<-0x1634 L=0x8000 H=0xc02c #32768 Send
Comsrv[8]:WM_DDE_DATA 0x1634->0x17f4 L=0x266 H=0xc02c 1 1 0 1 A Get
Comcli[9]:WM_DDE_DATA 0x17f4<-0x1634 L=0x2be H=0xc02c 1 1 0 1 A Get
Comcli[10]:WM_DDE_ACK 0x17f4->0x1634 L=0x8000 H=0xc02c #32768 Get
Comsrv[9]:WM_DDE_ACK 0x1634<-0x17f4 L=0x8000 H=0xc02c #32768 Get
Comcli[11]:WM_DDE_TERMINATE 0x17f4->0x1634 L=0 H=0
Comsrv[10]:WM_DDE_TERMINATE 0x1634<-0x17f4 L=0 H=0
```

notice that the display on the screen stops. Repeat the experiment with COMCLI/COMSRV, and observe that the typeout continues. Hence, if an application receiving real-time data cannot stand being suspended by a modal operation, use of a separate DDE server may help solve the problem.

Of course, a DDE application can generate quite a bit of overhead. As you will see, sending even a single character to the communications server involves creating, using, and disposing of several global data objects as well as the transmission and reception of two windows messages. Even so, the performance of COMCLI and COMSRV has been adequate at those baud rates supported by Windows.

How COMCLI and COMSRV Work Together

COMCLI and COMSRV together provide the user with a simple terminal program. Basically, COMSRV knows how to read from and write to the communications ports and how to set the port device control block (DCB). Its application name is "Comsrv". It can manage up to four topics—COM1, COM2, COM3, and COM4—if the hardware and software support are present (Windows currently supports only two communications ports). Finally, it responds to two items, **Get** and **Send**.

COMCLI understands how to display a byte stream in its window, accept keyboard input, manage the caret (screen cursor), and interact with the user to obtain basic communications parameters. Both programs understand enough of DDE to interact with each other. However, neither is an absolutely complete implementation, and, in fact, each program was written with the other in mind; consequently, there are some simplifications and omissions (intentional or otherwise). The most notable is that communications error conditions are generally ignored, and certain messages and their accompanying global objects are not tracked very well.

Assuming both programs are running in Windows, a typical session begins when COMCLI is requested to connect to the server. A dialog box is presented, from which the user can select communications parameters. For simplicity, COMCLI is limited to COM1 or COM2 and only a few values of baud rates, parity types, and byte sizes. If "OK" is chosen, an INITIATE message is broadcast with application name "Comsrv" and the port name as the topic. If the port is available, COMSRV opens it and acknowledges; in response, COMCLI executes a command to set the port's DCB. If the set is successful, COMCLI requests to be advised whenever new data arrives at the port.

COMCLI can send keystrokes to the port by poking it to the server. If COMCLI is minimized to an icon, it unlinks from COMSRV; it resumes the link when the window is restored.

The session continues until either COMSRV or COMCLI is closed or the user disconnects. The user can switch ports only by terminating the current session and beginning a new one. It is mildly annoying that baud rate, byte size, and parity can be changed only in the same way.

If the user has more than one communications port, a second session can be initiated by running another instance of COMCLI. Internally, COMSRV handles multiple sessions by creating a child window for each port it discovers. Each instance of a client talks to one of these children. Normally iconic, if COMSRV is restored to normal size, you will see the children displayed along the left side of the window, along with the hex value of their window handles.

A number of improvements can be made to these programs. For example, it would be useful to be able to set new communications parameters without disconnecting the session. The client may want to change the size of the port input and output queues, which currently are fixed by the server. Smart-terminal emulation in the client would be more useful. Finally, the client has no access to other data items supplied by the communications interface, notably the event mask. This suggests a second execute command to set the mask (which could be sent at the same time the device control block is set) and an additional item about which the client should be advised. In this case, the update sequence for the mask would probably be handled best as an alarm (deferred update).

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C++ Tools

An object-oriented library to help you learn and use C++!

Full source code is included, so you can see how C++ works, while you improve your productivity!

• The ***Rogue Wave C++ Toolbox*** is a set of C++ classes that can dramatically simplify your programming! Now you can make powerful, maintainable code that is still as fast and efficient as handcrafted C. • ***Vectors and Matrices:*** Add two vectors as easily as two doubles! Or read an array without knowing its size! Vectors and matrices of type int, float, double, complex, signed and unsigned char. Optional *bounds checking*. • ***Operators and Functions:*** All of the standard C arithmetic and assignment operators (such as "+" and "+=") and math functions (cos(), *etc.*), have been extended to include arrays and *complex numbers*. • ***Linear Algebra:*** Fast matrix inversion and solutions of linear equations. • ***Fast Fourier Transform*** server classes allow you to take the FFT or inverse FFT of 1- and 2-D arrays quickly and easily - perfect for image analysis! • ***Slice and Pick operators*** allow all operations to be performed on only selected elements, rows, columns or submatrices. • ***Much, much more!***

$150

$3 shipping
Visa/MC accepted

- ***Full Source Code!***
- 180 page manual full of C++ hints and tips
- Numerous examples and test programs
- No Royalties
- Compiles with Zortech and most Unix compilers

Rogue Wave

P.O. Box 85341
Seattle, WA 98145
(206) 523-5831

Number **324** on the Reader Service Card

A professional workstation environment for OS/2™

Hamilton C shell™

The superior alternative to the standard OS/2 command processor. Faithfully recreates and updates the entire C shell language as described in the Berkeley 4.3 *UNIX® Programmer's Manual*. Created explicitly for OS/2 using modern incremental compiler technology. Not one line ported from or created on anything but OS/2. Very high performance. Extensive support for multi-threading.

Features: Very long command lines: 64KB = 10x the UNIX limit • Fully recursive filename wildcarding: complex patterns, multiple directory-level matches • Fully nestable `if`, `foreach` and other control structures • Variables, arrays and a wide range of expression operators and built-in functions • History for recalling and editing past commands • Faster interrupts, I/O redirection and piping • Command substitution: use the output of one command as command-line arguments to another • Aliases and shell procedures • Filename completion • Arrow keys • Hashing to speed searches of the `PATH` directories • Background threads and processes, `ps` and `kill` • Numerous professional-quality utilities.

Requires OS/2 1.1 or later. All executables will run properly inside a Presentation Manager window. Not copy-protected.

$350.00. Unconditional satisfaction guarantee.

Phone 508-358-5715. MasterCard and Visa accepted.

Hamilton Laboratories
13 Old Farm Road, Wayland, MA 01778–3117

Number **334** on the Reader Service Card

Announcing ...

PC-lint 4.0

The Best Lint Just Got Better

PC-lint is a C source code analysis tool. It will check code more thoroughly than a C compiler and, because it looks across multiple modules, it enjoys a perspective that your compiler doesn't have.

Control your development time and cost. Find C source code errors before they find you.

Over **270 error messages**. More than **90 options** for complete customization. Suppress error messages, locally or globally, by symbol name, by message number, by filename, etc. **Easy to Use** - If you know C you already know how to use PC-lint. Check for **portability** problems. Alter size of scalars. Adjust format of error messages. Automatically generate ANSI **prototypes** for your K&R functions.

New Features in Version 4.0

- **Lint Object Modules** - incremental linting - big linting.
- **Weak Definial Checking** - macros, structs, enums, typedef's and declarations not used globally and locally. Header files not used. Externals that can be made static. Declarations that can be moved from headers. etc.
- **Library Headers** - avoid redundant prototype processing.
- **Over 70 new messages and 30 new options,** including identifier truncation clashes.
- **New "Elective Notes" message category.**
- **Unix lint compatibility**

Mainframe and Mini Programmers

Get **FlexeLint 4.0** (formerly Generic Lint) in shrouded source form for VAX/VMS, Unix, IBM VM/MVS, QNX OS-9, DG AOS/VS, etc. Requires only K&R C to compile but supports ANSI. Pricing starts at $798. Call for details.

Gimpel Software

3207 Hogarth Lane Collegeville PA 19426
(215) 584-4261 - CALL TODAY
(215)584-4266 - FAX
PC-lint is only $139 - Specify MS-DOS or OS/2
30 Day Money-back Guarantee

Quantity and educational discounts available. PA add 6% sales tax. Outside USA add $20.
PC-lint, FlexeLint and Generic Lint are trademarks of Gimpel Software.

Contact this advertiser directly.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

At the implementation level, a more robust system for tracking DDE messages is sorely needed. In our programs, the basic policy for handling global data objects is to request the recipient to dispose of them, and we suppose that this will always occur. Generally, this is a good assumption as long as the locking of data handles succeeds. If the locks fail, however, global objects may begin to accumulate in memory, since the handling of ACK messages is currently very rudimentary in both the client and the server. A number of related issues are also not addressed—for example, knowing which messages are being acknowledged. It appears that the only way to achieve a really foolproof system is to introduce a state machine, complete with time-out mechanisms, to control the session.

Viewing DDE Messages

Both programs on the *PJ* code disk have been linked with a module called DDEPRINT so that DDE messages can be sent to **stdout**; simply redirect Windows to the device with the command **Win** (or **Win386**) **>device**. The most useful choice is to a secondary monitor using the device driver OX.SYS, which really *must* be standard equipment for any Windows developer. (A copy of OX.SYS is supplied on the code disk; install it with **device=ox.sys**. Send any I/O to **stdout**, such as **printf**, by redirecting Windows to AUX).

If you don't have an auxiliary terminal or a secondary monitor, define WINAUX in DDEPRINT and make each both server and client. Before running either, execute the WINAUX.EXE program (see *PJ* 7.2), which is also included on this issue's code disk. Now you should see the DDE messages being displayed in the WINAUX window.

Assuming you have the required hardware and software in place, you may view the messages simply by choosing the corresponding menu item. Note, however, that COMSRV has all user selections attached to the system menu since the server normally runs as a minimized window, in which only this menu is easily visible.

A typical message display has the following fields, all on one line:

- Application name with next message number in brackets
- The DDE message in the form WM_DDE_XXX
- Sending window handle -> receiving window handle (for outbound messages)
- Receiving window handle <- sending window handle (for inbound messages)
- Low (prefixed by L=) and high (prefixed by H=) words of *lParam*

The remaining information in the line is message dependent. Included are strings from atoms (which may not exist, in which case they are displayed as integers prefixed with "#"), execute strings, status words, flags, and data strings. As much detail as possible is supplied.

The box on page 54 shows a sample session in which the client connects at 1200 baud, 8-bit bytes, no parity, and 1 stop bit to COM1, pokes the letter *A* to the port, receives the echo from the port, and terminates.

If you have access to Microsoft Online or other bulletin boards carrying Windows programs, you can download XLDDEKIT.ARC. This file contains a nice application called DSPY.EXE, which also allows you to view DDE messages. It does not depend on any sort of links with the target programs and can be used for viewing all DDE messages in the system.

The printing of DDE messages is hidden in a routine called **DDEPostMessage**, which is used also to dispatch the message. Called with the same parameters as **PostMessage**, the message parameters are also sent to **DDEPrint** if the appropriate flag has been set.

The Client Program, COMCLI

COMCLI is somewhat similar to the WNTERM program (*PJ* 7.6). COMCLI has a single, teletypewriter-style window that displays the most recent text on the bottom line and scrolls the preceding lines up. However, the routines in WNTERM needed to read, write, and set the communications ports have been replaced by those that send and receive DDE messages.

For example, instead of the communications-port device control block being set directly, a command string is assembled using the recommended syntax and sent as a WM_DDE_EXECUTE message to the server. Similarly, in response to keystrokes, the characters are packaged as a WM_DDE_POKE message to COMSRV. Finally, COMCLI receives information to be displayed on the screen via a WM_DDE_DATA message.

Listing 1 provides some of the interesting routines from the client program. Note that a session begins when the user activates the Connect menu and selects a port and its parameters (top of listing). At this time, a global variable called *hWndServer* is null. A WM_DDE_INITIATE message is then broadcast by a call to **DDEInitiate**, and, although this function does not itself modify *hWndServer*, some sort of magic has apparently affected it: If its value changes, the menu and window title are altered, and a call is made to set communications parameters. So, even before the menu handler has completed its processing, a number of interesting events have occurred.

The magic is a DDE exchange between COMCLI and the server. It is important to note that **DDEInitiate** *sends* a broadcast to all other windows in the system and does not return until *all* interested windows have answered the inquiry. The answers come back as WM_DDE_ACK messages and are processed by the function **DDEAck**. If one is seen to contain the desired application and topic atoms, a suitable server has been found, and *hWndServer* is set to its window handle. Eventually, **DDEInitiate** returns, and the remaining functions associated with the menu activation continue.

In this program, the server and topic atoms have been explicitly specified. If we had made the topic null, then the server would return as many messages as there were ports available; therefore, it is imperative that unwanted topics be terminated, since the server opens the corresponding port for *each* topic ACKed. Without the termination message, the unused ports would remain open and would be unavailable to other applications or clients. Note that a null topic might be practical when it

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Introducing C talk / Views™

Reduce C Programming for MS Windows by 75%

Introducing C_talk/Views™, a major productivity tool for Microsoft® Windows program development in C. Slash your MS Windows program development time by up to 75% through the powerfull object-oriented programming (OOP) technology of C_talk. With C_talk/Views, the most common elements of all MS Windows programs are packaged into reusable software components-object classes. Over forty software components are included as a complete applications development framework. These object classes provide an easy-to-learn and powerfully consistent interface to the rich functionality of MS Windows. C_talk/Views is also a real OOP environment. Use the software components as they are, or extend them to meet the most demanding applications.

Coming soon...C_talk/Views for the Apple Macintosh®... now your MS Windows program will port to the Mac with little or no change!

$450.00 In the US, add $5 for shipping. ($25 for Int'l handling)

CNS, Inc.
Software Products
7090 Shady Oak Rd.
Minneapolis, MN 55344
612-944-0170, Fax 612-944-0923

CNS®

. . . providing and advancing object-oriented methodology.

Contact this advertiser directly.

4c™ *Hypertext* Source Code Databank for Programmers

Explore source code! Navigate uncharted logic trails easily!

4 c includes:

★ **A Source Code Analyzer:**
The analyzer maps and cross-references functions and variables to their respective path and file names. The cross-reference table can be displayed alphabetically and by type of variable. It allows a programmer to edit functions directly without specifying a file name.

★ **A Point-and-Shoot Zoom Key:**
Which instantly locates any function, global variable, structure definition, #define statement, or macro, etc., in any file, directory, and drive, and displays it in a new edit window.

★ **A Full Featured Built in Editor:**
Featuring multiple files, multiple windows, cut-and-paste between files, search-and-replace, auto indent, tab support, and a familiar user interface with pull-down menus, or use your own editor!

"Given a procedure's identifier, an editor should be able to bring up the full procedure interface, including parameters, so that you don't have to remember how many and in what order. Wouldn't it be nice to hit a hot key on the name of a complicated data type and have its definition pop up in a window? It may seem like a real small matter, but how many of you still flip through printouts doing things the editor should be doing?"

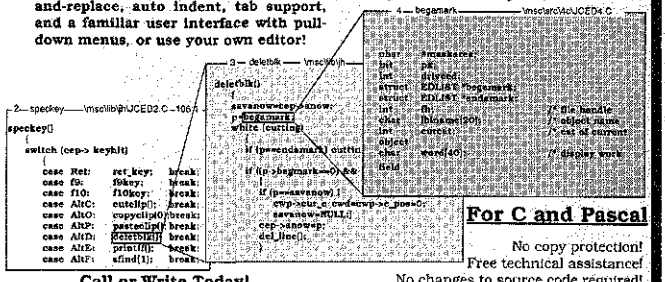
- Phillipe Kahn, "The Connection", Spring/Summer 1988

4 c does this and more

"4c's Magic is apparent..."
- PC Magazine, February 1989

"Daylight in the C Confusion"
- Toolbox magazine, December 1988

"...Destined to become the standard editor for C..."
- Computer Personlich magazine, October 1988



For C and Pascal

No copy protection!
Free technical assistance!
No changes to source code required!
Required IBM PC compatible with 350K RAM.
30 Day Money Back Guarantee!
Only $119.00 plus shipping
Visa, Mastercard, check, money order, or COD.

Call or Write Today!

Tri-Technology Systems Inc.
1225 S. Elgin, Forest Park, Il. 60130
Phone: 312-366-7595

Number **276** on the Reader Service Card

"To the rescue... the C Toolbox offer a number of solid utilities to reformat, analyze and validate source code...."
---Bill Rubin, The C Gazette, Winter 1988

The C Programmer's Toolbox contains more than 20 tools for the serious C programmer. The tools are available in three different development environments: PC/MS DOS; Macintosh's MPW; and Sun's UNIX. Some of the major tools include:

CDecl	Translates C declaration statements to/from English *(helps you decipher those complicated C variable declarations and construct new ones). {MPW & Sun only}*
CFlow™	Determines how a program is organized and what is in the runtime library *(quickly shows you function hierarchy and interdependancies)*
CHilite™	Highlights and prints your C source code *(great for creating your final documentation). {MPW only}*
CLint™	Syntactically checks one or more of your C source code files *(use this to find those hard to find, syntactically correct programming mistakes)*
Cpp	Preprocesses C source code files that have ANSI macro definitions *(helps you port ANSI C code to/from your target environment). {MPW & Sun only}*
CPrint™	Reformats/beautifies your C source code *(makes code easy to read and ends disputes on source code formats).*
CritPath™	Determine a program's critical path. *(helps you improve your program's performance) {PC DOS only}*
CXref™	Cross references and checks usage of up to 17 different classes of symbols in your C source code *(cross references any size program and number of listings)*
PMon™	Monitors a program's execution *(non-invasively monitors your program's function execution, system service and BIOS usage). {PC DOS only}*

"... The tools I tried operated both quickly and flawlessly..."
---M. Steven Baker, Programmer's Journal, May 1988

C™ Programmer's Toolbox / MPW PC Sun

The tools include support for ANSI C, Microsoft C, Turbo C, Quick C, Sun UNIX C, Macintosh's MPW C, THINK C and more. The Toolbox easily handles any size program through its virtual memory system. Several tools also include a C macro processor that accepts K&R, ANSI and other C dialects.

C Programmer's Toolbox™
MPW ($295), PC Volume I or II ($99.95),
PC Volume I & II ($175), Sun ($495)
C Programmer's McTool Series™ (Macintosh - non-MPW)
McCPrint™ ($59.95), McCLint ($99.95)

"... it's a complete set of utility programs which can make the life of a C programmer easier... I really like this package..."
- Scott Robert Ladd, Micro Cornucopia, November 1988

Call and Order today...
(CA residents, please add sales tax; Overseas shipping extra)

MasterCard VISA®

MMC AD Systems™
Box 360845
Milpitas, California 95035
(408) 263-0781

Number **94** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

is desirable for the server to cycle through all available ports to find one available.

To set the communications parameters, a command must be executed in the server. DDE commands have a prescribed syntax, which in this case is realized in the form

```
[SetDCB("baud,wordsize,parity,stopbits,...")]
```

SetDCB is the opcode, and the operand is a string of integers separated by commas. A value and its position in the string are interpreted by the server as a field in the port's device control block data structure; however, if there is no integer between two adjacent commas, that particular field remains unaltered.

In COMCLI, the user can select the first three fields from a dialog box. The number of stop bits is then determined by the baud rate, and all other fields are fixed. In particular, note that software handshaking is used with the usual values for the Xon/Xoff characters. Also, there is no mechanism for selecting the outbound- and inbound-queue sizes for the ports; the server currently uses 512 bytes for the receive queue and 128 bytes for the send queue. A more sophisticated program would allow the user a wider choice of values.

If the EXECUTE message fails, the session is terminated; otherwise a link is established with a call to **DDEAdvise**. The link remains active until **DDEUnAdvise** is called, either from the menu's Disconnect command or when the window is made iconic (as detected by receipt of the WM_SIZE message).

When the link is in effect, **DDEData** processes new information from the server. First the sender's handle, item, and format are checked; if acceptable, the data is copied into a local buffer and displayed on the screen. Finally, an ACK is returned, since the server cannot send any more information until receipt is acknowledged. Without the ACK, the client may miss data, because either the server never stops the data flow from the host or the client's message queue is overrun.

In response to a WM_CHAR message, **DDEPoke** is called with the character and the repeat count. The data is then passed to the server via DDE. Usually the data is just one character. Observe the effort, however, in just getting the character out the door; the corresponding effort by the server to recover and transmit it is equally arduous.

The session terminates either when the client or the server is closed or the user chooses to disconnect. A series of WM_DDE_TERMINATE messages are then exchanged. In particular, the function **DDETerminate** is executed only when a termination message has been received from the server.

The Server Program, COMSRV

COMSRV begins its initialization phase by registering two window classes, one for the main window and the second for managing DDE sessions. Then, when the main window is created, a call is made to test for the presence of ports COM1 through COM4. For each port found, a data structure **DDECOMStat**, indexed by the port number, is initialized, and an instance of the DDE window class is created as a child of the main window. Each newly created child is assigned the communications-port index as its identifier. Later, when a DDE message is received, the ID is retrieved and used to identify the session's data structure. Some of the details are shown at the top of Listing 2.

When a WM_DDE_INITIATE message is received from a possible client, **DDEInitAction** is called. if the topic is one that can potentially be supported, an attempt is made to open the corresponding communications port. If the open call is successful, an ACK message containing the handle of the associated child window is returned. At this point, the session has been established.

Once the port parameters have been set by a WM_DDE_EXECUTE command from the client, the server is prepared to receive data as a WM_DDE_POKE command and write it to the communications port. Note that a number of checks are made to verify that the item and the client handle are as expected and that the data is in the CF_TEXT format before it is written. Only then is the status word set to show success for return by the ACK message.

Anticipating the need to poll the communications port, a basic **PeekMessage** loop is used to retrieve and dispatch Windows messages. If an ADVISE link has been established, the appropriate communications port is read, and the data is sent to the interested client. A simple looping mechanism is used to cycle through each of the ports having active sessions to share the resources of the CPU.

When a WM_DDE_TERMINATE message arrives from the client, the affected port is closed and **DDEComStat** is reset. If COMSRV itself is terminated by a WM_CLOSE or WM_ENDSESSION message, all ports are examined; if one is open, it is closed. In the case of a WM_CLOSE message, the affected client also receives a TERMINATE message.

Parsing the Execute String

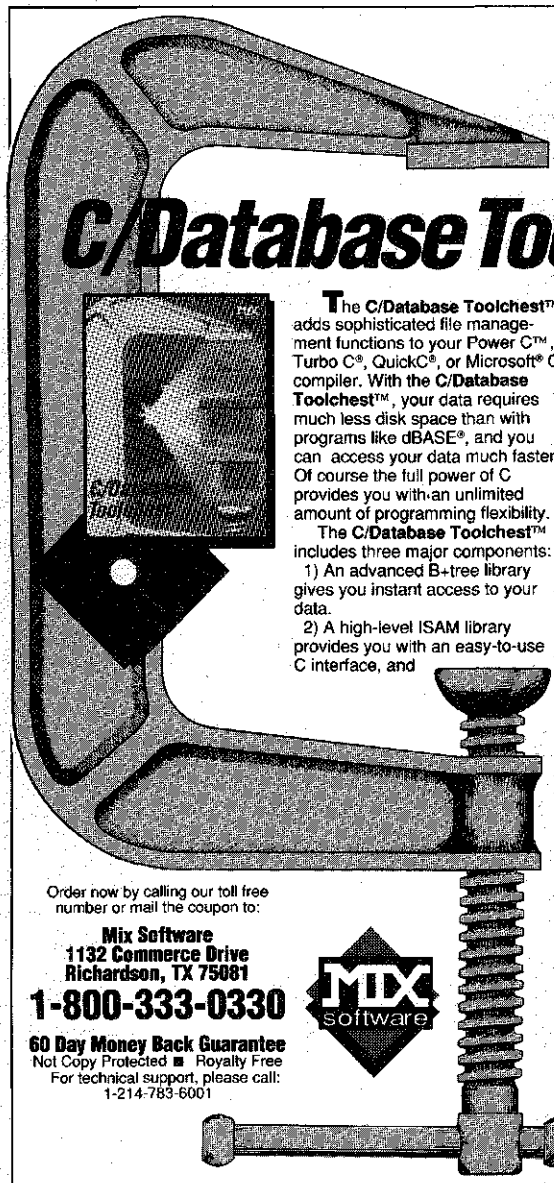
The DDE specification provides a fairly complete syntax describing the format of strings used to execute commands in the server. We have already seen the one we use to set the DCB, and in the earlier article, we described the grammar and gave some examples. In general, a command string is a sequence

```
[command1][command2] ...
```

where each command consists of a single opcode and possibly one or more operands. The operand list is enclosed in parentheses and delimited by commas, or the operand is a string, as in our application. Both opcode and operand are implementation specific.

In COMCLI, it is relatively easy to form the execute string expected by the server, and all such strings should generally be quite easy to assemble. It is quite another matter to parse and execute them in the server in a fairly general way. We found the means in two well-known programs from the UNIX world, Lex and Yacc, which have been adapted to MS-DOS by Mortice Kern

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Turn your favorite C compiler into a powerful database manager with the

C/Database Toolchest

The **C/Database Toolchest™** adds sophisticated file management functions to your Power C™, Turbo C®, QuickC®, or Microsoft® C compiler. With the **C/Database Toolchest™**, your data requires much less disk space than with programs like dBASE®, and you can access your data much faster. Of course the full power of C provides you with an unlimited amount of programming flexibility.

The **C/Database Toolchest™** includes three major components:

1) An advanced B+tree library gives you instant access to your data.

2) A high-level ISAM library provides you with an easy-to-use C interface, and

3) A complete database manager (with C source code included) shows you how to create impressive applications.

You also receive a comprehensive 350 page manual and a utility for converting dBASE® files.

The **C/Database Toolchest™** supports features that you'd expect to find only in products costing ten times as much. Advanced features include variable length records, variable length keys, multiple keys per index, and multiple indexes stored in a single file. Your data files can contain an unlimited number of records, and each record can be as large as 32K bytes in length.

About the only thing that the **C/Database Toolchest™** doesn't do is cost you a lot of money. We've kept our price low so you can manage your budget as easily as your data.

Now Only $19.95!

Order now by calling our toll free number or mail the coupon to:

Mix Software
1132 Commerce Drive
Richardson, TX 75081

1-800-333-0330

60 Day Money Back Guarantee
Not Copy Protected ■ Royalty Free
For technical support, please call:
1-214-783-6001

MIX software

Order Coupon

Name_____
Street_____
City_____
State_____ Zip_____
Telephone_____
Paying By____ Money Order____ Check
____ Visa ____ MC ____ AX ____ Disc.
Card#_____
Exp. Date_____
Disk Size____ 5¼" ____ 3½"

Qty.	Product	Price	Subtotal
___	C/Database Toolchest	$19.95	____
___	C/Database Library Source, B+tree & ISAM library source code	$10.00	____
	Add Shipping ($5 USA, $20 Foreign)		____
	Texas Residents Add 8% Sales Tax		____
	Total Amount of Your Order		____ J

C/Database Toolchest and Power C are trademarks of Mix Software. QuickC and Microsoft C are registered trademarks of Microsoft. Turbo C is a registered trademark of Borland. dBASE is a registered trademark of Ashton-Tate.

Number 266 on the Reader Service Card

Confusing Code?

```
#include <stdio.h>
text_count(){int c,nlines,nwords,nchars,inword;
inword=NO;nlines=nwords=nchars=0;while((c
=getchar())!=EOF){++nchars;if (c=='\n')
++nlines;if ((c==' ')||(c=='\n'))inword=NO;else
if(inword==NO){inword=YES;++nwords;}}printf(
"%d %d %d\n",nlines,nwords,nchars);}
```

C It Your Way!

```
#include <stdio.h>

text_count ()
{
     int c, nlines, nwords, nchars, inword;

     inword = NO;
     nlines = nwords = nchars = 0;
     while ((c = getchar()) != EOF) {
          ++nchars;
          if (c == '\n')
               ++nlines;
          if ((c == ' ') || (c == '\n'))
               inword = NO;
          else if (inword == NO) {
               inword = YES;
               ++nwords;
          }
     }
     printf ("%d %d %d\n", nlines, nwords, nchars);
}
```

NOW with **C-Clearly™**, format C source code exactly the way you want it. C-Clearly's context sensitive analysis will format any C program in your own personal or corporate style.

EASY to use, C-Clearly's style templates are a snap to modify, since they resemble C source code you edit into your preferred format. Templates are included for several common styles as well as standard K & R.

LISTINGS can also be created with function names and comments highlighted for improved readability. Listing options include line numbers, headers and/or footers and flow lines.

IDEAL for making obtuse code clear. Allows all of your source code to be presented in a consistent format of your choosing. Also great for code walkthroughs and final documentation listings.

WORKS with all IBM PC, XT, AT, PS/2 and compatibles, with 512K RAM. Automatically processes all include files and pre-processor statements. ANSI-C compatible. Not copy protected.

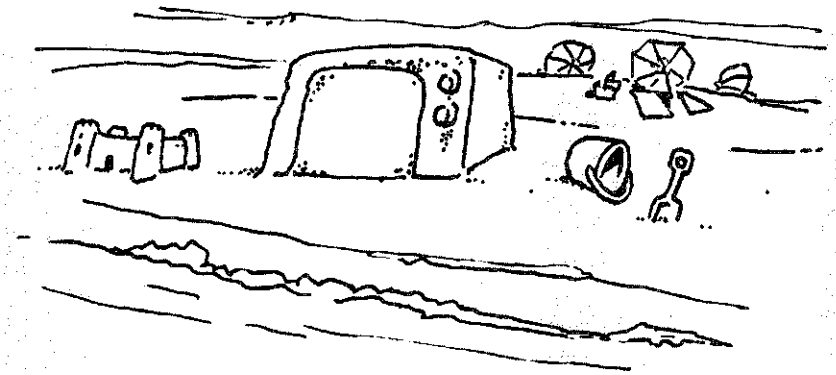
C-Clearly ***$129.95***

Shipping & Handling USA $3; Canada/Mexico $10; Other Countries $15; CA Residents add sales tax. Visa/MasterCard/COD accepted.

For orders or information call:

1-800-662-8266

V COMMUNICATIONS, INC.
3031 Tisch Way, Suite 802, Dept. PJ5
San Jose, CA 95128 (408) 296-4224



Number 341 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Systems. However, Lex and Yacc are designed basically to read and write input, output, and errors in ways that are at variance with Windows. Several modifications of the I/O interface are needed to remove calls to **getc**, **putc**, **printf**, and so on. Fortunately, the MKS versions provide ways to make the necessary alterations.

The Lex Module

Listing 3 contains the main features of the Lex module used in COMSRV. Following a definition of the integer class and the format of the only opcode of interest, several basic Lex functions and parameters are redefined. For example, the input stream is a null-terminated string located in the Windows global heap. Hence **yygetc**, which normally calls **getc**, is redefined to obtain simply the next character in the string until its end is reached, whence EOF is returned. Similarly, **output**, which normally uses **putc** to display unmatched output, is redefined to do nothing.

YY_FATAL, which is executed if the look-ahead buffer overflows, is changed to show a message box and do a clean Windows exit. Some extra efficiency in processing is achieved by setting YY_INTERACTIVE to 0.

Finally, something must be done to prevent Lex from jamming after receiving its first EOF. By design or oversight, MKS Lex does not restart very well. However, after some experimentation, we found that by defining YY_PRESERVE, we can call Lex again, even after seeing an EOF. Thus, it becomes possible to use Lex, not only in this application, but in defining terminal-emulation-state machines in Windows as well.

In COMSRV, Lex is used to recognize the opcode **SetDCB** and an integer. All other characters are simply returned to the caller (the parser, **yyparse**), whether they are tokens or not. Hence, unrecognized input is detected, not by Lex, but by Yacc to centralize any error response.

The Yacc Module

As with Lex, any code that writes I/O needs to be modified. For Yacc, only **yyerror** has to be redefined. In this case, it does nothing, since we decided to defer all error reporting to the client. As an aid to debugging, the command string is displayed in a message box if the parse does not succeed. Such failures can happen for lexical reasons (unrecognized token), because of errors in syntax, or if the device control block cannot be set.

Listing 4 shows the main elements of the Yacc module needed in COMSRV. The grammar itself is general enough to handle a sequence of command strings, although only one is needed here. Specializations have been made for the opcode and the operand string, but it should be fairly easy to modify or extend the grammar to other needs.

Linking with EXCEL

COMSRV will form a link with Microsoft Excel; what follows is the simplest way to demonstrate it: If you have only one communications port (say, COM1), you will need a separate terminal connected to that port and set at 9600 baud, 8 bits, no parity, and 1 stop bit. Run Windows, COMSRV, and Excel, in that order. In some cell of a blank worksheet, type the formula "=Comsrv|COM1!Get". If the link is successful, an "N/A" message will be displayed. Now, each time you type a character on the terminal, it will appear in the cell. It is interesting to view the DDE messages on the auxiliary monitor, if your machine is so equipped.

If you have two ports, you can try running COMCLI against COMSRV on one port at 9600 baud and looping the results back into Excel via a null modem cable on the other.

It is possible to write Excel macros to send more-complicated sequences of DDE messages. We have not experimented along these lines, but with some effort, you may be able to carry out other commands, such as EXECUTE and POKE.

Conclusion

The programs presented above show both the usefulness and the complexity of DDE. Because it is a standard, however, DDE provides the best means for automatically accessing data in other applications that have provided the necessary functions. DDE also affords an improved method of obtaining multithreaded behavior, but at the cost of additional transport overhead.

DDE has a degree of independence from the using application. Hence, if you give careful attention to detail, you should be able to write a set of support routines or libraries serving any number of applications and simplifying the implementation of DDE in a particular program. PJ

William S. Hall is a senior engineer at Novell, in San Jose, California. He specializes in Windows software.

Code follows.

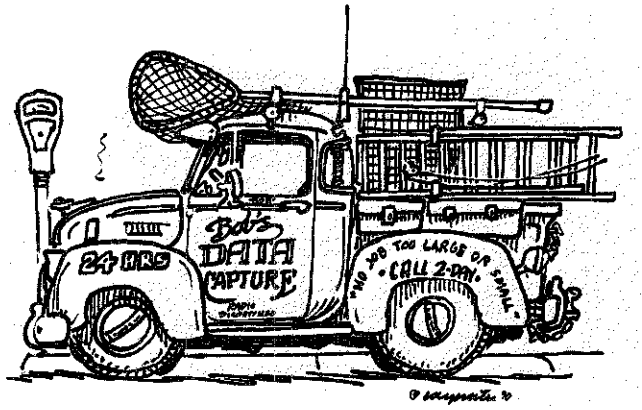
References and Vendor

Hall, W. S., "Dynamic Data Exchange In Windows," *Programmer's Journal* 8.1 (1990), 46–54.

Hall, W. S., "A Simple Terminal Program for Windows," *Programmer's Journal* 7.6 (1989), 26–38.

Hall, W. S., "A Simple Debugging Utility for Presentation Manager and Windows," *Programmer's Journal* 7.2 (1989), 56–73.

Mortice Kern Systems (MKS), 35 King Street North, Waterloo, Ontario N2J 2W9, Canada.



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Code continues . . .

Code from "Communications Using Dynamic . . . ," by William S. Hall

PJ CODE

All code in this issue is available through our bulletin board services or directly from *PJ* on disk. See pages 8, 88, 22, and 93.

LISTING 1

```
/* Excerpts from COMCLI DDE functions */

/* Menu items */
void NEAR WndCommand(HWND hWnd, WORD wParam)
{
    FARPROC fp;
    switch(wParam) {
        case IDM_CONNECT:          // connect to server
            fp = MakeProcInstance((FARPROC)GetCommPortDlg, hInst);
            if (DialogBox(hInst, MAKEINTRESOURCE(DT_GETPORT),hWnd,fp)) {
                DDEInitiate(hWnd);
                if (hWndServer) {
                    InSession = TRUE;
                    /* modify menu and title bar */
                    SetCommData(hWnd, hWndServer);
                } else
                    ShowMessage(hWnd,IDS_SERVERBUSY,MB_ICONEXCLAMATION | MB_OK);
            }
            FreeProcInstance(fp);
            break;
        case IDM_DISCONNECT:       // disconnect from server
            if (ShowMessage(hWnd,IDS_DISCONNECTOK,MB_ICONQUESTION | MB_YESNO)
                    == IDYES)
                DDEPostMessage(hWndServer, WM_DDE_TERMINATE, hWnd, 0L);
            break;
    }
}

/* broadcast initiate message */
static void NEAR DDEInitiate(HWND hWnd)
{
    char buf[BUFSIZE];
    ATOM aTopic, aServ;

    GetAtomName(CommData.aPort, buf, sizeof(buf));
    aTopic = GlobalAddAtom(buf);
    aServ = GlobalAddAtom(szServer);
    SendMessage((HWND)(-1),WM_DDE_INITIATE,hWnd,MAKELONG(aServ,aTopic));
    GlobalDeleteAtom(aServ);
    GlobalDeleteAtom(aTopic);
}

/* act upon ACK message from server */
void NEAR DDEAck(HWND hWnd, HWND hServer, WORD a1, WORD a2)
{
    char buf1[BUFSIZE], buf2[BUFSIZE];

    if (!InSession) { /* ACK of initiate messages.  Accept or terminate */
        GlobalGetAtomName(a1, buf1, sizeof(buf1));
        GlobalGetAtomName(a2, buf2, sizeof(buf2));
        if ((aServer == FindAtom(buf1)) && (CommData.aPort == FindAtom(buf2)))
            hWndServer = hServer;
        else
            DDEPostMessage(hServer, WM_DDE_TERMINATE, hWnd, 0L);
        GlobalDeleteAtom(a2);
        GlobalDeleteAtom(a1);
    }
```


UNDERSTANDING
C
VIDEOTAPES

Learn the techniques used by experienced C programmers.

Get guaranteed results from this informative 3-volume, 6-hour videotape course. Loaded with examples that cover every nook and cranny of ANSI C. Handy reference manual included.

$59.95 postpaid (plus tax in Texas). Money-back guarantee. No extra charge for C.O.D. orders.

To order C.O.D., call:
800-762-5515

(Canadian orders: $64.95 in U.S. funds)

AUTOMATION EDUCATION
2309 Royce Drive
Arlington, TX 76016-1232

Number **207** on the Reader Service Card

C Programmers
Explore a world of DOS memory
Beyond 640K

Use the PowerSTOR C library to exploit:
- LIM 4.0 or LIM 3.2 Expanded Memory
- AT Style Extended Memory
- Hard Disk Space
- Any Combination

- PowerSTOR is the easy way to use system memory resources. It self-configures at run time, under control of your software.
- PowerSTOR makes system memory resources available to your software as STORheaps. STORheaps can be allocated and de-allocated dynamically. Up to a gigabyte of PowerSTOR memory can be used as one huge STORheap or many smaller STORheaps.
- PowerSTOR uses none of the normal heap and it minimizes its usage of stack space. These valuable resources are preserved for your use.

Library with K & R standard source code — only $92.50
(includes shipping)

Order today and explore beyond the limits.

Acme Software

P. O. Box 40734, Portland, Oregon 97240 — Phone (503) 235-2816

Number **305** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Hall code, continued

2

```
    else if (hWndServer == hServer) {
        if (a2 == hCommand) {  /* ACK execute command, terminate if failed */
            if (a1 & 0x8000)
                DDEAdvise(hWnd, hServer);
            else {
                /* display message box with command string to user */
                DDEPostMessage(hServer, WM_DDE_TERMINATE, hWnd, 0L);
            }
            GlobalFree(hCommand);
        } else ... /* ACK other messages */
    }
}

/* prepare command string from data in dialog box */
static void NEAR SetCommData(HWND hWnd, HWND hServer)
{
    LPSTR lptr;
    char buf[100], tbuf[BUFSIZE];

    hCommand = GlobalAlloc(GMEM_MOVEABLE | GMEM_DDESHARE | GMEM_ZEROINIT,
                           (LONG)sizeof(buf));
    if (lptr = GlobalLock(hCommand)) {
        LoadString(hInst, IDS_SETDCB, tbuf, sizeof(tbuf));
        sprintf(buf,"[%s(\"%u,%u,%u,%u,,,,%u,,,,,,%u,%u,,%u,,,,%u,%u,%u,%u\")]",
                                    tbuf,
                                    CommData.BaudRate,
                                    CommData.ByteSize,
                                    CommData.Parity,
                                    CommData.StopBits,
                                    1,                  // binary mode
                                    1,                  // xon-xoff out
                                    1,                  // xon-xoff in
                                    1,                  // no null chars
                                    17,                 // xon char
                                    19,                 // xoff char
                                    32,                 // xonlim
                                    32                  // xofflim
                );
        lstrcpy(lptr, buf);     /* should check length of buffer here */
        GlobalUnlock(hCommand);
        DDEPostMessage(hServer, WM_DDE_EXECUTE, hWnd, MAKELONG(NULL, hCommand));
    }
}

/* set up a link to get updates from the server */
void NEAR DDEAdvise(HWND hWnd, HWND hServer)
{
    HANDLE hAdvise;     /* this handle should be tracked */
    DDELN FAR *pAdvise;
    char buf[BUFSIZE];
    ATOM aItem;

    if (hAdvise = GlobalAlloc(GMEM_MOVEABLE | GMEM_DDESHARE | GMEM_ZEROINIT,5L)) {
        if (pAdvise = (DDELN FAR *)GlobalLock(hAdvise)) {
            pAdvise->fAckReq = 1;       // handshake
            pAdvise->fDeferUpd = 0;     // supply data instead of alarms
            pAdvise->fRelease = 1;      // release data object
            pAdvise->cfFormat = CF_TEXT;  // use this format
            GlobalUnlock(hAdvise);
            GetAtomName(CommData.aGetItem, buf, sizeof(buf));  // item
            aItem = GlobalAddAtom(buf);
            DDEPostMessage(hServer,WM_DDE_ADVISE,hWnd,MAKELONG(hAdvise,aItem));
        }
    }
}
```

3

```
/* terminate a link with the server */
void NEAR DDEUnAdvise(HWND hWnd, HWND hServer)
{
    char buf[BUFSIZE];
    ATOM aItem;

    GetAtomName(CommData.aGetItem, buf, sizeof(buf));
    aItem = GlobalAddAtom(buf);
    DDEPostMessage(hServer,WM_DDE_UNADVISE,hWnd,MAKELONG(0,aItem));
}

/* process data received from the server.  Ignore alarms (null data handle) */
void NEAR DDEData(HWND hWnd, HWND hServer, HANDLE hData, ATOM aItem)
{
    DDEUP FAR *pData;
    char buf[BUFSIZE];
    BYTE *rgb;
    int len;
    WORD wStatus = 0;

    GlobalGetAtomName(aItem, buf, sizeof(buf));
    if ((hWndServer == hServer) && (CommData.aGetItem == FindAtom(buf))) {
        if (pData = (DDEUP FAR *)GlobalLock(hData)) {
            if (pData->cfFormat == CF_TEXT) {
                len = lstrlen(pData->rgb);
                if (rgb = (BYTE *)LocalAlloc(LPTR, len)) {
                    lstrcpy(rgb, pData->rgb);
                    HideCaret(MWnd.hWnd);
                    TTYDisplay(&MWnd, len, rgb);
                    SetCaretPos(MWnd.Pos.x, MWnd.Pos.y);
                    ShowCaret(MWnd.hWnd);
                    LocalFree((HANDLE)rgb);
                    wStatus = 0x8000;
                }
            }
            GlobalUnlock(hData);
            if ((pData->fRelease) && (wStatus))
                GlobalFree(hData);
        }
    }
    if (pData->fAckReq)
        DDEPostMessage(hServer, WM_DDE_ACK, hWnd, MAKELONG(wStatus, aItem));
    else
        GlobalDeleteAtom(aItem);
}

/* send characters to the server */
void NEAR DDEPokeChar(HWND hWnd, BYTE ch, WORD count)
{
    HANDLE hPoke;       /* this handle should be tracked */
    DDEPOKE FAR *pPoke;
    register int i;
    char buf[BUFSIZE];
    ATOM aTopic;

    if (hPoke = GlobalAlloc(GMEM_MOVEABLE | GMEM_DDESHARE | GMEM_ZEROINIT,
                            (LONG)(count+5))) {
        if (pPoke = (DDEPOKE FAR *)GlobalLock(hPoke)) {
            pPoke->fRelease = 1;
            pPoke->cfFormat = CF_TEXT;
            for (i = 0; i < count; i++)
                pPoke->rgb[i] = ch;
            GlobalUnlock(hPoke);
            GetAtomName(CommData.aSendItem, buf, sizeof(buf));
            aTopic = GlobalAddAtom(buf);
            DDEPostMessage(hWndServer,WM_DDE_POKE,hWnd,MAKELONG(hPoke,aTopic));
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
        }
    }
}

/* terminate a DDE session and reset menus and title bars */
void NEAR DDETerminate(HWND hWnd, HWND hServer)
{
    char buf[BUFSIZE];

    if (hWndServer == hServer) {
        hWndServer = NULL;
        InSession = FALSE;
        ShowMessage(hWnd, IDS_SERVEREXIT, MB_OK | MB_ICONASTERISK);
        ModifyMenu(hWnd, IDM_DISCONNECT, IDM_CONNECT, IDS_CONNECT);
        LoadString(hInst, IDS_TITLE, buf, sizeof(buf));
        SetWindowText(hWnd, buf);
        if (IsWindow(hServer))
            DDEPostMessage(hServer, WM_DDE_TERMINATE, hWnd, 0L);
    }
}
```

LISTING 2

```
/* Excerpts from COMSRV */

/* take care of any additional initialization details */
void FAR WndInit(HWND hWnd, LONG lParam)
{
    register int i, cid;
    HMENU hMenu;
    char buf[40];

    /* increase the size of the message queue */
    SetMessageQueue(32);
        ...

    /* try to open ports; fill in data base
     * and open a DDE window for those available */
    for (i = 0; i < MAXPORTS; i++) {
        DDEComStat[i].cid = INT_MIN;
        LoadString(hInst, IDS_COM1 + i, buf, sizeof(buf));
        cid = OpenComm(buf, INQUESIZE, OUTQUESIZE);
        if ((cid != IE_BADID) && (cid != IE_HARDWARE)) {
            DDEComStat[i].hDDEWnd = MakeAndShowChildWnd(hWnd, i);
            DDEComStat[i].aPort = AddAtom(buf);
        }
        if (cid >= 0)
            CloseComm(cid);
    }
}

/* respond to WM_DDE_INIT message */
void NEAR DDEInitAction(HWND hClient, ATOM aApp, ATOM aTopic)
{
    char buf[40];
    register int i;

    if (aApp) {          // Non-Null app, so see if it is ours
        GlobalGetAtomName(aApp, buf, sizeof(buf));
        if (aAppName != FindAtom(buf))
            return;
    }
    if (aTopic == NULL) {       // Reply on all possible topics
```



```
        for (i = 0; i < MAXPORTS; i++) {
            if ((DDEComStat[i].hDDEWnd) && (DDEComStat[i].hWndClient == NULL)) {
                GetAtomName(DDEComStat[i].aPort, buf, sizeof(buf));
                OpenThePort(DDEComStat[i].hDDEWnd, hClient, buf, i);
            }
        }
    }
    else {              // Reply only to the one requested
        GlobalGetAtomName(aTopic, buf, sizeof(buf));
        for (i = 0; i < MAXPORTS; i++) {
            if ((DDEComStat[i].hDDEWnd) && (DDEComStat[i].hWndClient == NULL)) {
                if (DDEComStat[i].aPort == FindAtom(buf)) {
                    OpenThePort(DDEComStat[i].hDDEWnd, hClient, buf, i);
                    break;
                }
            }
        }
    }
}

/* open the comm port; if successful, ACK the initiate message */
static void NEAR OpenThePort(HWND hWnd, HWND hClient, char *buf, int index)
{
    register short cid;
    ATOM a1, a2;

    if ((cid = OpenComm(buf, INQUESIZE, OUTQUESIZE)) >= 0) {
        DDEComStat[index].cid = cid;
        DDEComStat[index].hWndClient = hClient;
        a1 = GlobalAddAtom(szAppName);
        a2 = GlobalAddAtom(buf);
        SendMessage(hClient, WM_DDE_ACK, hWnd, MAKELONG(a1,a2));
    }
}

/* send a character string to the port */
void NEAR DDEPoke(HWND hWnd,HWND hClient,HANDLE hData,ATOM aItem,WORD index)
{
    char buf[40];
    DDEPOKE FAR *lptr;
    register WORD wStatus = 0;

    GlobalGetAtomName(aItem, buf, sizeof(buf));
    if ((aSendItem == FindAtom(buf)) &&
                            (DDEComStat[index].hWndClient == hClient)) {
        if (lptr = (DDEPOKE FAR *)GlobalLock(hData)) {
            if (lptr->cfFormat == CF_TEXT) {
                /* write to the port, but ignore any errors */
                WriteComm(DDEComStat[index].cid,lptr->rgb,lstrlen(lptr->rgb));
                wStatus = 0x8000;
            }
            GlobalUnlock(hData);
            if (lptr->fRelease)
                GlobalFree(hData);
        }
    }
    DDEPostMessage(hClient, WM_DDE_ACK, hWnd, MAKELONG(wStatus, aItem));
}

/* Read and process windows messages, if any.
 * Otherwise, cycle through the available comm ports.
 * If fAdvise is TRUE, poll the port and dispatch the data. */
static void NEAR DoPeekMessage()
{
    MSG msg;
    static int index = 0;
```

Code continues . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Hall code, continued

6

```
    if (PeekMessage((LPMSG)&msg,NULL,0,0,PM_REMOVE)) {
        if (msg.message == WM_QUIT)
            exit((int)msg.wParam);
        TranslateMessage((LPMSG)&msg);
        DispatchMessage((LPMSG)&msg);
    }
    else {
        if (index < MAXPORTS) {
            if ((DDEComStat[index].hDDEWnd) && (DDEComStat[index].fAdvise))
                if ((DDEComStat[index].fAck && DDEComStat[index].AdvData.fAckReq)
                        || (DDEComStat[index].AdvData.fAckReq == 0))
                    ProcessComm(index);
            index += 1;
        }
        else
            index = 0;
    }
}

/* examine the comm port, and if necessary, dispatch data to client */
static void NEAR ProcessComm(int i)
{
    COMSTAT ComStatus;
    register int num;
    DDEUP FAR *pData;
    char buf[40];
    register ATOM aTopic;

    /* see if any data present */
    GetCommError(DDEComStat[i].cid, (COMSTAT FAR *)&ComStatus);
    if ((num = ComStatus.cbInQue) == 0)
        return;

    num = min(num, INQUESIZE);
    GetAtomName(aGetItem, buf, sizeof(buf));
    aTopic = GlobalAddAtom(buf);
    /* if update is deferred, send no data */
    if (DDEComStat[i].AdvData.fDeferUpd)
        DDEPostMessage(DDEComStat[i].hWndClient, WM_DDE_DATA,
                       DDEComStat[i].hDDEWnd, MAKELONG(NULL,aTopic));
    /* otherwise read the comm port and send back the information */
    else {
        if (hData = GlobalAlloc(GMEM_MOVEABLE | GMEM_DDESHARE | GMEM_ZEROINIT,
                        (LONG)(INQUESIZE + sizeof(DDEUP)))) {
            if (pData = (DDEUP FAR *)GlobalLock(hData)) {
                pData->fAck = 0;
                pData->fRelease = 1;
                pData->fAckReq = DDEComStat[i].AdvData.fAckReq;
                pData->cfFormat = DDEComStat[i].AdvData.cfFormat;
                /* read the port, but ignore any read errors */
                ReadComm(DDEComStat[i].cid,pData->rgb, num);
                GlobalUnlock(hData);
                DDEComStat[i].fAck = FALSE;
                DDEPostMessage(DDEComStat[i].hWndClient, WM_DDE_DATA,
                        DDEComStat[i].hDDEWnd, MAKELONG(hData,aTopic));
            }
        }
    }
}
```

7

LISTING 3

```
/* excerpts from the Lex module for the Windows DDE Communications server. */

setdcb [sS][eE][tT][dD][cC][bB]          /* only opcode we recognize */
integer [0-9]+
%{

extern LPSTR cmdstr;             /* execute command string from client */

/* not interactive */
#undef YY_INTERACTIVE
#define YY_INTERACTIVE  0

#define YY_PRESERVE       /* this seems to allow Lex to restart after seeing EOF */

/* define our own input */
#undef yygetc
#define yygetc() mygetc()

/* don't need this one */
#undef output
#define output(c)

/* take care of fatal errors a la windows */
#undef YY_FATAL
#define YY_FATAL(msg) { MessageBox(hWndMain, msg, szAppName, \
                                   MB_OK | MB_ICONEXCLAMATION); \
                        PostMessage(hWndMain, WM_SYSCOMMAND, SC_CLOSE, 0L); }
%}

%%
{setdcb} {
    return SETDCB;
}

{integer} {
    yylval = atoi(yytext);
    return INTEGER;
}

. {
    return *yytext;   /* let yacc handle any unexpected input */
}
%%

/* read the command string */
static int NEAR mygetc()
{
    register ch = *cmdstr++;
    return(ch ? ch : EOF);
}
```

LISTING 4

```
/* Excerpts from the Yacc module for the Windows DDE Communications server. */

%{
static int portnum;
static int cmd;
static int dcbbuf[MAXSIZE];
static int dcbndx = -1;

/* define our own error message handler */
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
#undef yyerror
#define yyerror(msg) myerror(msg)
%}

%token '[' ']' '(' ')' '"' ','
%token SETDCB
%token INTEGER

%%
exestring : exestring command
          | /* NULL */
          ;
command : '[' opcodestr ']' { /* execute command */
                              if (cmd == SETDCB) {
                                  cmd = 0;
                                  if (SetDCB())  /* failed, show error */
                                      YYABORT;
                              }
                            }
          ;
opcodestr : opcode
          | opcode parameterlist
          ;
parameterlist : '(' parameters ')'
              ;
parameters : parameter
           | parameters ',' parameter
           ;
parameter : '"' quotedop '"'
          ;
opcode : SETDCB { /* initialize command type and dcbbuf parameters */
                  cmd = SETDCB;
                  for (dcbndx = 0; dcbndx < MAXSIZE; dcbndx++)
                      dcbbuf[dcbndx] = -1;
                  dcbndx = -1;
                }
       ;
quotedop : field
         | quotedop ',' field
         ;
field : INTEGER { /* enter this value into dcbbuf */
                  if (dcbndx++ < MAXSIZE) dcbbuf[dcbndx] = $1;
                }
      |         { /* empty field, so just bump pointer */
                  dcbndx++;
                }
      ;
%%
/* handle the execute string from the client */
void FAR DDEExecute(HWND hWnd, HWND hClient, HANDLE hCommand, WORD index)
{
    WORD wStatus = 0;
    portnum = index;

    if (cmdstr = GlobalLock(hCommand)) {
        wStatus = (yyparse() == 0 ? 0x8000 : 0);
        GlobalUnlock(hCommand);
    }
    DDEPostMessage(hClient, WM_DDE_ACK, hWnd, MAKELONG(wStatus, hCommand));
}

myerror(msg)
{
    ; // do nothing
}
```


WHICH LANGUAGE DO *YOU* SPEAK??

MXW

Mixed Language Windowing System

1 MXW version is compatible with C, QC, Pascal, Assembler, MS BASIC, QB, and FORTRAN

$59.95

(FREE demo disk available)

mtd
SOFTWARE
development company, ltd.
21 Ralph Ave., N. Brentwood, NY 11717

(516) 434-1979

Number **312** on the Reader Service Card

Multitasking

Do the multitasking systems you are considering using seem a bit like over kill or need an awful lot of memory? Maybe all you really need is a system which can run more than one DOS program concurrently. If this sounds like your situation, consider MultiDos Plus. For $24.95 or $99.95 (if you want our source code) you can run up to 32 tasks and still use MS-DOS or PC-DOS. For more information please call or write for a free brochure.

Nanosoft
800-678-2141
508-651-0091
FAX 508-655-8860
13 Westfield Road
Natick MA 01760

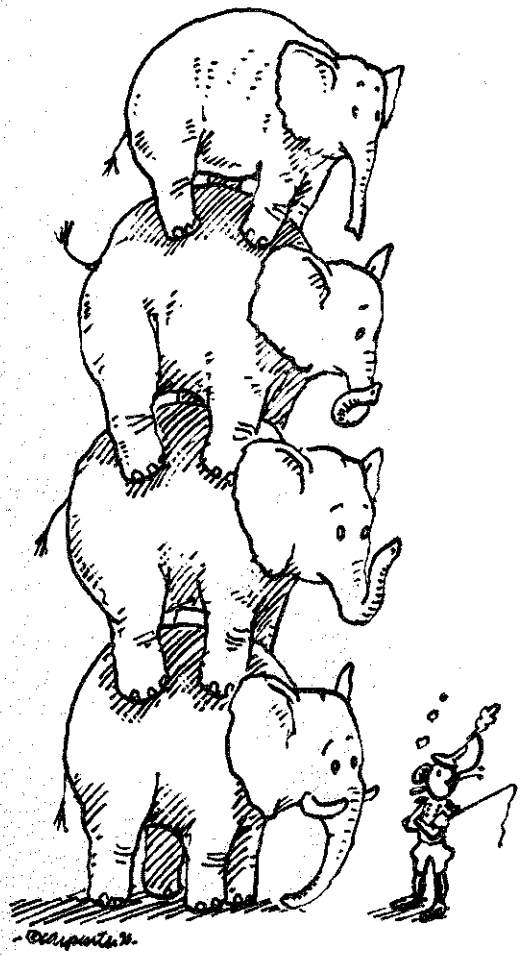
Number **109** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Manipulating Large Data

by

John Otken



In previous columns, I present a convention for local labels within the body of an assembly language procedure. Because the labels are formed from the first three initials of the words in the procedure name and a numeric suffix (for example, abc1, abc2, abc3), I call them "three-initial labels." The biggest drawback to this method is the angst that occurs when indices end up out of sequence: I waste countless hours restoring label order when I should be programming.

In this column, I present a new program that automatically resequences local labels in a source file. I use the opportunity to finish start-up code and introduce include files.

Start-Up Code

The start-up code in Listing 1 has several improvements from the code in my preceding column. The first changes are necessary to move the code into a separate module that becomes part of our assembly language library. To use the start-up code, declare it external and call it from main. A program needs only one **dosseg** directive to "do the right thing" while linking; placing the directive here is one less thing to worry about in the main program.

I move the NULL segment declaration to the start-up module. You may recall that this segment is necessary for aligning DGROUP to a 16-byte paragraph boundary.

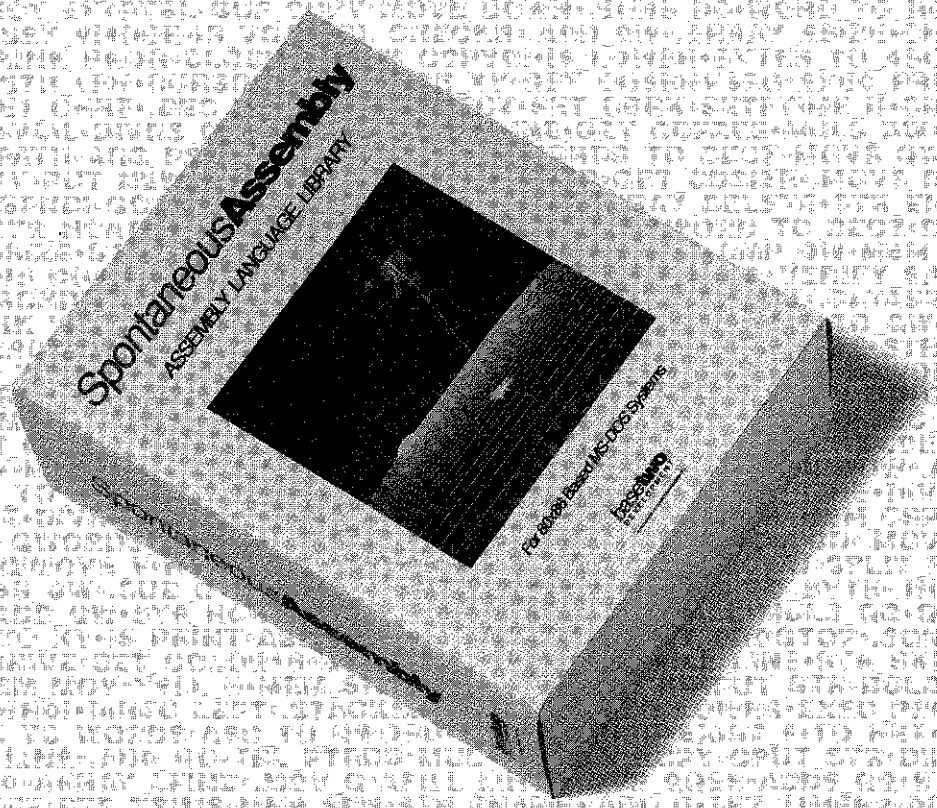
```
NULL segment para public 'BEGDATA'
NULL ends
```

I also add code to zero the uninitialized data segment that belongs to class BSS. Two special symbols—**_edata** and **_end**—help find this segment. These symbols, though poorly documented, are useful: **_edata** marks the start of uninitialized data in BSS, and **_end** marks the end of uninitialized data and the start of the stack. Without these symbols, one would have to define similar symbols and pay close attention to module ordering. Using this built-in feature of the linker is easier and less problematic.

The start-up code now stores the DGROUP segment in a static variable, **dgroup_segment**. (DGROUP is the collection of initialized and uninitialized data segments and the stack segment. The linker keeps the three segments separate but contiguous. To the program, DGROUP appears as one big segment, still less than 64K.) This permits you to load the DGROUP frame into a segment register with a single instruction:

```
mov  ds,dgroup_segment[bp]
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Serious Assembly.

Nobody really takes assembly language seriously. Nobody, that is, except Lotus, WordPerfect, Novell, and everyone else who needs the tightest, fastest code possible.

But assembly is tough . . . unless you have plenty of ready-to-use assembly language routines at your fingertips. That's why industry leaders spend a lot of time and money creating their own, proprietary assembly language libraries.

You could spend your valuable development time building a library; or you can use **Spontaneous Assembly**, the complete **assembly language library** specifically designed for serious assembly language development.

Over 600 functions and macros.

- String manipulation • Memory manipulation
- Near/far/relative heap management
- Doubleword/quadword integer math
- Array management, sorting, and searching
- Character/numeric/string conversion • Buffer management • Date and time manipulation
- File and directory management • File I/O
- Keyboard I/O • Program/environment control
- Critical error management • Screen I/O with windowing—and more. All carefully documented, tested, and ready to use.

Tight, fast code. Fast.
Every routine is hand-coded and hand-optimized (in assembly, of course). And a consistent, register-oriented parameter-passing convention makes these routines remarkably easy to use. With Spontaneous Assembly, coding in assembly is as fast as coding in a high-level language—without the overhead.

Powerful memory model support.
Spontaneous Assembly unleashes the full power of memory models in assembly language. It supports all Microsoft/Borland standard memory models as well as custom models and mixed-model programming. And it gives you complete control over segment/group names and attributes.

Extensive documentation.
A complete reference manual describes every function, macro, and variable in detail. Step-by-step instructions and technical notes explain memory models, library customization, integration with C, and much more.

Spontaneous Assembly.
Serious assembly language programming doesn't need to be hard any more. Now it can be Spontaneous.

See for yourself. Call now to take advantage of our introductory price. Then use Spontaneous Assembly for 60 days. If you're not 100% satisfied, send it back for a full refund!

Introductory Price.
$199 Plus Shipping & Handling.
Full source code included. No royalties.
60 day money-back guarantee.

Visa • MasterCard • Discover

Call now!

1-800-ASSEMBLY

Orders • Information • Support

basetwo DEVELOPMENT

For 80x86–based systems running DOS 2.0 or later. MASM 5.1 or TASM 1.0 recommended.
Base Two Development • A Division of Acclaim Technologies, Inc. • 11 East 200 North • Orem, Utah 84057 • (801) 222-9500.
Spontaneous Assembly and Base Two Development are trademarks of Acclaim Technologies, Inc.

Number 343 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
mov  ds,ax      ;  the old way
```

You can push SS and pop into DS or ES, but I find this sequence difficult to read. This is one instance when a macro can increase readability. For now, I prefer the move from **dgroup_segment**.

```
push  ss        ; set DS to DGROUP the
pop   ds        ;  hard-to-read way
```

The start-up code from the preceding column sets BP to the offset of symbol **dgroup_start**, which happens to be 0. I set BP that way so that you could use a nonzero BP and take advantage of short negative offsets from BP.

```
mov  bp,offset dgroup_start
```

This approach does not offer sufficient benefits in return for added complexity. BP is now 0. You can freely define static variables in the **.data** and **.data?** segments without concern for the position of the module that defines **dgroup_start**.

If you need additional performance after completing development, you set up the global BP structure described in *Programmer's Journal* 7.6 (November/December 1989). You must ensure that the BP structure expands into the start of DGROUP.

Static Variable Insurance

If we know BP is 0, we can take advantage of an assembler feature to protect against program bugs. This bug appears when you forget to apply the [BP] index to a static-variable reference and DS does not contain DGROUP.

```
mov  cx,label_count  ; "[bp]" is missing bug
```

The assembler generates a simple move instruction that uses the DS segment register instead of SS. The code does not work correctly, but it looks right—such bugs can take hours to find. The easy way to insure against this bug is to instruct the assembler to make no assumptions about the contents of DS. Use the following **assume** statement:

```
assume ds:nothing
```

When the assembler encounters a DGROUP memory reference without a [BP] index, it forces an "SS:" override prefix, because SS is the only segment assumed to contain DGROUP. The .model small directive implies the following **assume** statement:

```
assume cs:_TEXT,ds:DGROUP,ss:DGROUP
```

This is one exception to my rule against automatic overrides; it's reliable insurance against an error that is easy to make. This tip does not help if you forget to code the **assume** statement. To guard automatically against this error and provide many other conveniences, we have include files.

Include Files

An include file is a mechanism for incorporating a file's contents into a source file. Assembly language modules commonly include a file with convenient symbol and macro definitions. The name of the include file given in Listing 2 is asm.inc.

The first line, .xlist, turns off the list-file generation (include files only waste paper in a listing). The second line sets the page size for MASM listings. While MASM assumes a prehistoric paper width of 80 columns, TASM has a reasonable default and OPTASM has several configuration options for listing control.

The next two lines select the small-memory model and remove assumptions for the DS register. The **assume** statement protects against bugs should you forget the [BP] index in an instruction.

The next line can force the linker to search your assembly language library automatically. Because a comment character begins this line, the assembler ignores the **includelib** directive. If you want to use this feature, remove the semicolon.

```
bptr  equ  byte ptr  ; abbreviate type overrides
wptr  equ  word ptr
dptr  equ  dword ptr
```

These three lines define shorthand symbols for type-override operators. You must use a type override with instructions that accept immediate and untyped memory operands.

```
mov bptr [si],0  ; set delimiter
```

If a register type does not match a memory-operand type, an override is required. The common example occurs when storing a dword pointer into a variable. The 80X86 CPU has a single instruction to load a 32-bit pointer, but it lacks a corresponding store instruction.

```
   .data
file_pointer dd ?
   .code
...
   mov  wptr file_pointer[bp],si   ; must use two lines
   mov  wptr file_pointer[bp+2],ds ;  to store pointer
```

Instructions with type overrides are usually long. Long lines are difficult to read and can mess up the alignment of comments. These abbreviations keep instruction length to a minimum.

The next line is a FILENAME_MAX equate borrowed from UNIX for the maximum length of filenames. Other miscellaneous equates for control characters and a NULL_POINTER follow. These equates increase program readability.

Definitions for six macros are next. These macros duplicate OPTASM features. Four of the macros permit more-compact EXTRN directives by eliminating the type from each argument. For example, the EXTN macro

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

accepts a list of near external procedures. You do not have to append ":near" to any of the arguments.

```
extn   strskp_white,skip_line   ; compare these 2 lines
extrn strskp_white:near,skip_line:near
```

The last two macros permit pushing or popping of multiple registers. Multiline lists of push or pop waste valuable screen space during editing; these macros help you write more attractive and readable code. Compare the following two fragments. The push and pop instructions obscure the rest of the code in the second sample.

```
pushm di,si,es                  ; first sample
les   di,output_pointer[bp]
call  strcpy
popm  es,si,di
ret

push  di                        ; second sample
push  si
push  es
les   di,output_pointer[bp]
call  strcpy
pop   es
pop   si
pop   di
ret
```

These macros are unnecessary in OPTASM, and since the include file uses conditional assembly, OPTASM ignores the definitions. In this case, the benefits outweigh the evils of macros and conditional assembly.

Renumbering Program

ORDER is a program that renumbers the local labels in assembly language procedures. The program accepts a limited syntax because I do not want to write an entire assembler.

Exercise care when you use this program, because it can introduce bugs into your code. For example, if you jump to a local label from another procedure, ORDER might change the label but not the jump instruction. A good way to verify proper operation is to assemble your program first, ORDER one or more source files, and reassemble. If the binaries do not compare, check for errors.

After startup, ORDER reads the entire source file into memory and allocates an output bucket for the updated source. The program is ready to process individual procedures.

The main loop begins at label "mail", and the program searches for the next procedure. It identifies a procedure by a line that begin with one or more alphanumeric characters followed by blank space and the PROC keyword. When it finds a procedure, it copies unprocessed source lines to the output bucket and then searches for the procedure end using a similar algorithm and the ENDP keyword. After finding a procedure body, the program makes multiple passes over it:

Count_proc_lines counts the number of lines in the body. If it contains too many or too few lines, the procedure is ignored. This protects the label table from overflow. Subsequent passes use the line count to figure out where a procedure ends.

Get_proc_initials searches for a local label formed from three letters and some digits. If it finds one, it records the three initials so that similar labels can be identified in the procedure.

Find_local_labels examines each line for a local label and records its numeric index in the **label_table**. If it finds labels in a format other than "three initial" (or 1$, 2$, and so on), the procedure is ignored.

The indices in **label_table** are binary, and there is no binary difference between the indices in xyz001, xyz01, and xyz1. The assembler can differentiate among these labels, but ORDER cannot; so, **atoi_special** detects leading zeros and returns an error. The error passes through **extract_label_number** and **find_local_labels**, and the procedure is ignored.

Check_label_sequence verifies label indices. If the labels are in proper order, resequencing is not necessary, and control reverts to the main loop.

Resequence_proc_labels scans each line in the procedure body. At the start of each line and after blanks or a comma, it looks for a local label. If the label syntax is good, it searches for the index in the **label_table** and substitutes the new index. **Resequence_proc_labels** stores the updated procedure body to the output bucket as it processes each line. Before branching back to the main loop, the program writes the procedure name to the console.

When the program reaches end-of-file, it checks whether any resequencing occurred. If not, the program exits; if so, ORDER renames the original .asm file to .bak and writes the updated file contents to disk.

ORDER uses the **read_entire_file** routine from the previous column. This limits input file size to about 63K but simplifies buffer management. I like to think of the benefits of this limitation—it offers an incentive to program modularity.

Manipulating Large Data

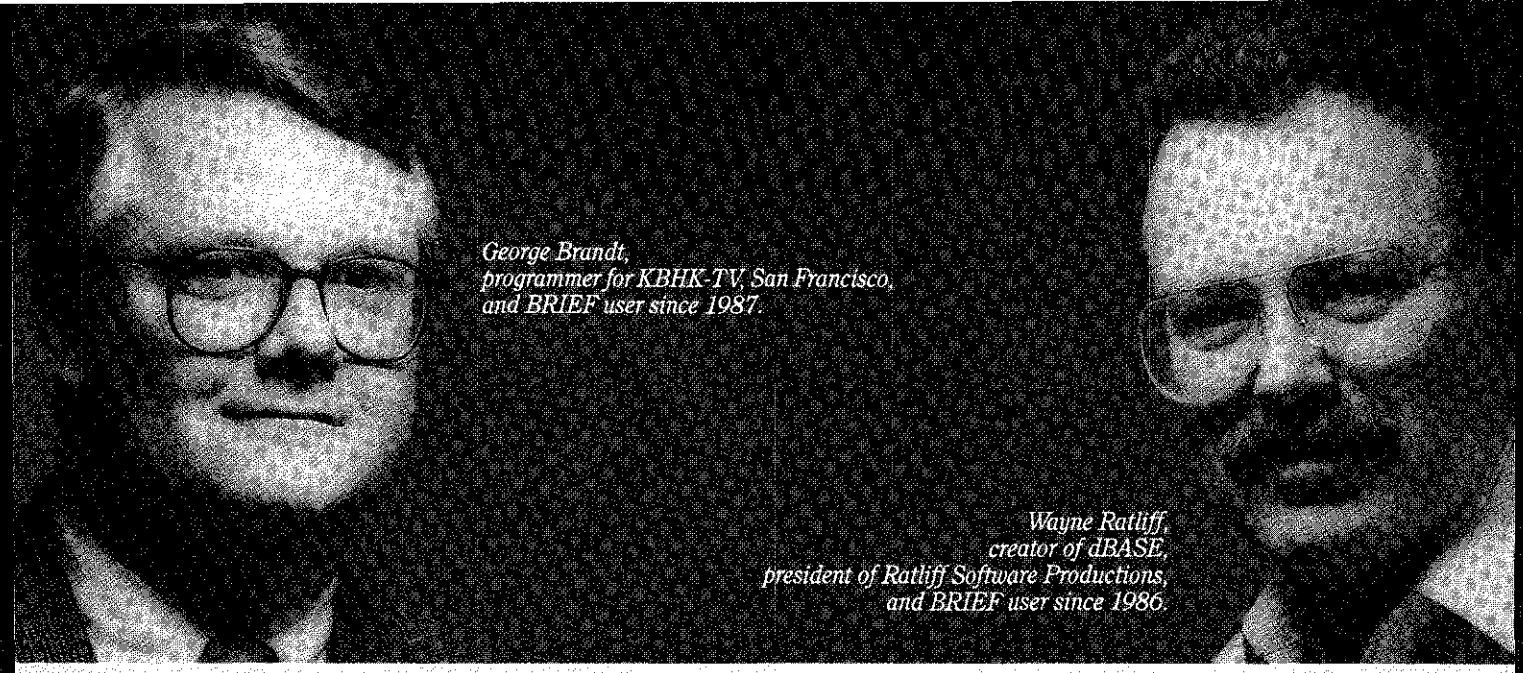
An interesting aspect of this program is its consistent use of registers. From **read_entire_file** through **update_asm_source**, DS:SI always points at the input file. Likewise, ES:DI is an output-file pointer (except in **convert_label_index**). CX is a count of either lines or labels in a procedure; only AX, BX, and DX are general purpose. I deliberately wrote the program this way, and the result is much easier to understand and debug.

There are two techniques for proficient large-data programming with the 80X86. The first is to design algorithms around the index registers—SI, DI, BP, and BX. They are the keys to effective use of the three "data"-segment registers—DS, ES, and SS.

You must access memory via a segment register. Smart programmers take advantage of instructions that use segment registers other than DS. The first example is static-variable storage in DGROUP. SS holds the DGROUP segment. If an instruction accesses static variables via SS, the contents of DS and ES are not important.

Because indexing off BP uses SS, BP is the key to DGROUP. That is why I set BP to 0 (indexing off 0 does

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



George Brandt,
programmer for KBHK-TV, San Francisco,
and BRIEF user since 1987.

Wayne Ratliff,
creator of dBASE,
president of Ratliff Software Productions,
and BRIEF user since 1986.

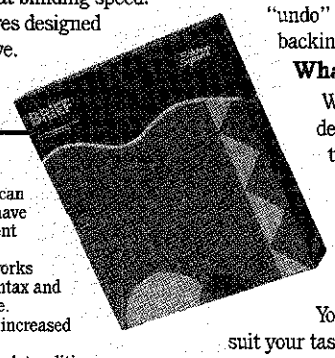
Of course they need completely different kinds of editors. That's why they both use BRIEF.

Let's face it. There's no such thing as two programmers who think alike. Or who work alike.

Because everybody's got their own style, their own habits. Their own quirky little requirements. As well they should. So, where are you going to find one editor that satisfies everyone? Simple. Look inside a BRIEF® box.

It's the right editor, right out of the box.

What you'll find inside is an enormously powerful, astonishingly easy, and highly-evolved program editor. One that works at blinding speed. And that's loaded with features designed to help you be more productive. One guaranteed to give you excellent results within one short hour of booting up the first time.



New BRIEF 3.0 highlights

- With our new C-like macro language, CBRIEF, C programmers can write macros on the fly and not have to switch between C and a different macro language.
- The new source level debugger works with both the original BRIEF syntax and the new CBRIEF macro language.
- Maximum lines per file has been increased to 4 billion.
- We've added to and improved template editing and smart indenting for ADA, Cobol, BASIC, FORTRAN, Modula-2, Pascal, and C.
- We've added editable, multiple keystroke macros that can be saved and restored.
- You can even save and restore your window set-ups.

Plus the features that made BRIEF famous.

- 300-level UNDO safety net
- Flexible windowing, unlimited quantity
- Unlimited number of files
- File size limited only by disk space
- Regular expression search and replace
- BRIEF's LISP-like macro language
- Toll-free tech support

You'll find features like automatic indenting, infinite windowing, and powerful regular expression searching that goes well beyond the ordinary text-matching you're probably used to.

You'll also find the features that made BRIEF famous. Including our near-miraculous UNDO, that doesn't just un-delete your last keystroke. It actually reverses the effect of any command. So you can explore, experiment, even make gigantic mistakes with hideous consequences, and just "undo" them. And then continue to undo, backing-up through 300 steps.

What you want is what you get.

What you'll also find is an editor that delights in adapting itself to you. With the simple, menu-driven SETUP program, you can quickly customize indenting style, support for any of 36 compilers, keyboard configuration, rate of keyboard autorepeat, and much more.

You can even edit BRIEF's macros to suit your taste. Or write your own in a snap.

Computer Language said that BRIEF is capable of "almost complete customization." And with new release 3.0, BRIEF is even more powerful and flexible.

For every programmer, it's a different BRIEF.

At last count, there were over 52,000 copies of BRIEF in the field, making life easier and working time more productive for all kinds of programmers—working in all kinds of languages. Programmers as different as Wayne Ratliff is from George Brandt. Which means there are over 52,000 different ways of working, and BRIEF fits them all.

Why not get BRIEF going to work for you? There's no risk. In fact, if you're not completely satisfied that BRIEF is exactly the editor you've been looking for, just return it within 30 days, and we'll return your money. No questions asked.

BRIEF 3.0

The Programmer's Editor

Solution Systems™
541 Main St., Suite 410
South Weymouth, MA 02190
FAX 617-337-7719

Try BRIEF at our risk. $199 plus $6 shipping (U.S. and Canada). 30 day money back guarantee.

Name _____
Company _____
Address _____
City _____ State _____ Zip _____
Phone _____
Paying by ☐ Check ☐ MC ☐ VISA
Card # _____ Exp. date _____
Signature _____
Disk size ☐ 5-1/4" ☐ 3-1/2" Format ☐ DOS ☐ OS/2

Mail this coupon or call toll-free today.

800-821-2492

Extension 445

Massachusetts & outside U.S. call 617-337-6963

International Distributors: England (0763) 73457 West Germany (069) 34 72 26 France 47.70.47.70 Australia (03) 663 6580 Denmark (02) 88 72 49 Norway 03-89 22 40 Austria 1 31 70 36 Italy (02) 837.83.41 Holland (020) 668 5606 Israel 4 238757

All product names are trademarks or registered trademarks of their respective owners. © 1989 Solution Systems

Number **47** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

not change the effective address). Unfortunately, you cannot double-index BX and BP. In the few instructions that use BX as a DGROUP index, an SS override appears.

The LODSB instruction makes SI the most efficient index register for reading character data. Most procedures that read from a character string should accept a pointer in DS:SI. Given that DS:SI is the "input" pointer for strings, it is also the input pointer of choice for almost any kind of data, although BX is almost as useful.

BX is a good second-choice index with any of the segment registers, and DS:BX is a useful base pointer to complex structures. As for DI, the STOSB instruction makes it the best choice for an output pointer.

When designing the entry and exit conditions of library routines, register choice is mostly automatic: Read a character string, use DS:SI; write a character string, use ES:DI.

The second programming technique for large data is to design algorithms that fit within the available register set of the 80X86 CPU. With only three "data"-segment registers, you cannot access data in more than three segments at once. If you ignore static variables and SS, you have only two segment registers left, DS and ES, and the natural way to work with them is to use DS for input and ES for output.

The 80X86 architecture is suitable for algorithms with a single input or a single output. If you can express program logic with this restriction, the code is much easier to write. The ORDER program is a great example: It reads one files, does some processing, and writes one file. The program was easy to write in assembly language.

Many programs can be expressed in terms of one-input–one-output logic. If you get stuck in a subroutine, your algorithm is probably too complex—break it into smaller components.

You can "bend" the one-input–one-output limitation. In **find_proc_endp**, I need a file pointer and a string pointer. Because the string was either "proc" or "endp", I could access the string pointer via SS:BX without using ugly instructions or bashing DI.

Some programs are so complex that they are not worth the effort to program in 80X86 assembly language—try C or another high-level language. That way, you can delegate the dirty work of debugging, maintaining, and documenting to a junior programmer!

Assembler News

Microsoft and Borland are working on new versions of their assemblers. I heard both products were in some stage of field testing back in November; if that is true, they should be ready when you read this. The major enhancement for both is OPTASM-like N-pass assembly, and OPTASM is two years old.

Why is Borland rewriting their assembler after only the first version? Perhaps they compiled a few 386 programs and noticed how gross the code looks. I hope the new TASM is more MASM compatible than the last version.

As for Microsoft, I predict MASM6 will be very late and full of bugs, because the top assembly language programmers have switched to OPTASM or TASM—there is no one left to test MASM. I said "no dice" to a MASM beta offer because most of my source code is no longer MASM compatible.

A Reference Revisited

You might recall (from an earlier article) my recommendation of *DOS Programmer's Reference*, by Terry Dettmann. Allen Wyatt of Que Corporation sent me the second edition (revised by Jim Kyle). Bravo!—an excellent book is even better. If you need the real technical nitty gritty of DOS, get this book.

You can download the code for this column off my bulletin board service or one of several *Programmer's Journal* bulletin boards. Or, for ten bucks, *PJ* will send you the code for the entire issue. If you call my BBS (213-559-1449), please leave me a comment with your views on this column. PJ

John Otken is the president of Soft Advances, where he develops software for embedded systems when he isn't snow-skiing. When the powder is deep at Mammoth, good luck reaching him at 10811 Washington Blvd., Suite 205, Culver City, CA 90232, 213-559-7015.

Code follows.

Performance Programming TOOLS

For IBM PC, XT, AT, PS/2 & compatibles

Commenting Disassembler

Sourcer™ creates detailed commented assembly source code and listings from memory and executable files. Built in data analyzer and code simulator produce great results. Supports 8088/87 through 80386/80387 and 80486 instruction sets. *"Sourcer is the best disassembler we've ever seen!" PC Magazine, January 17, 1989, page 101.* $99.95

BIOS Source

The **BIOS Pre-Processor™** with Sourcer provides the first means to obtain accurate legal source listings for any BIOS. Identifies entry points with detailed in-line comments, explains functions, and more! Requires Sourcer. $49.95; Sourcer & BIOS $139.95 (Save $10)

Unpack EXE Files

The **Unpacker™** group of programs assists with the debugging and analysis of software. Unpack packed EXE files and convert multi-segment COM files back to EXE. Includes handy interrupt viewer and video test program. $39.95.

Assembly Code Analyzer

ASMtool™ is an assembly language source code analysis tool that simplifies the understanding of complex software. Performs automatic flowcharting, analysis of stack and register usage, procedure tree diagrams, and static timing analysis. $89.95

Zoomable File Viewer

With **View-It™** zoom out to see up to 8 times more information on a single screen. View-It handles files of any size, provides dual windows, advanced search and mouse support, and is incredibly fast! EGA or VGA required for zoom feature. $69.95

Assembly Tools Package

ASM ProPak™ includes Sourcer with BIOS Pre-Processor, Unpacker, ASMtool and View-It. $299.75 (Save $50)

C Code Your Way

C-Clearly™ formats C source code according to your style preferences. Make your own style template or use and modify supplied templates such as standard K & R and others. Also produces formatted listings with optional flow lines, line numbers, headers/footers and more. $129.95

Shipping & Handling: USA: $5; Canada/Mexico $10; Other Countries $15; CA Residents add sales tax. Visa/MasterCard/COD accepted. If within 30 days of purchase you find our product does not perform in accordance with our claims, call our customer service department and we will gladly arrange a refund.

NOT COPY PROTECTED
30-DAY MONEY BACK GUARANTEE

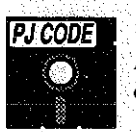
For orders or information call:

1-800-662-8266

V COMMUNICATIONS, INC.
3031 Tisch Way, Suite 802, Dept. PJ5
San Jose, CA 95128 (408) 296-4224

Number **125** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



All code in this issue is available through our bulletin board services or directly from *PJ* on disk. See pages 8, 88, 22, and 93.

Code from "Manipulating Large Data," by John Otken

1

LISTING 1

```
        title   start-up code for assembly language
        include asm.inc

        dosseg
NULL    segment para public 'BEGDATA'
NULL    ends

        public  psp,startup,dgroup_segment

        .data
psp                     dw      0
dgroup_segment          dw      0

        .data?
        extb    _edata,_end

        .code
        extn    set_argc_argv,ms_dos

;;      startup
;
;       entry   AX      offset stack
;               ES      program segment prefix
;               SS      DGROUP
;       exit    BP      0
;       may use AX,BX,CX,DX,DI,SI,DS,ES
;
startup proc
        cld
        mov     bp,0
        mov     psp[bp],es

        add     ax,15                   ; compute number of paragraphs
        mov     cl,4                    ;  in program
        shr     ax,cl
        mov     bx,ss
        add     bx,ax
        sub     bx,psp[bp]

        mov     ah,4Ah                  ; shrink program memory block for
        call    ms_dos                  ;  dynamic storage system

        mov     dgroup_segment[bp],ss

        mov     al,0                    ; zero bss segment
        mov     cx,offset DGROUP:_end
        mov     es,dgroup_segment[bp]
        mov     di,offset DGROUP:_edata
        sub     cx,di
        rep     stosb

        call    set_argc_argv
        ret
startup endp
```

2

```
        end
```

LISTING 2

```
        .xlist
        page    63,128

        .model  small
        ASSUME  ds:nothing
;       includelib lib

bptr                    equ     byte ptr
wptr                    equ     word ptr
dptr                    equ     dword ptr

FILENAME_MAX            equ     80

NULL_POINTER            equ     0

NULL_CHAR               equ     0
TAB_CHAR                equ     9
EOL_CHAR                equ     10      ; end of line
LF_CHAR                 equ     10      ; linefeed
CR_CHAR                 equ     13      ; carriage return
SPACE_CHAR              equ     ' '

        IFNDEF  .date   ; if not OPTASM
        .sall           ; don't show macro contents

extb    macro   a,b,c,d,e,f,g,h,i,j,k
        irp     xx,<a,b,c,d,e,f,g,h,i,j,k>
        ifnb    <xx>
        extrn   xx:byte
        else
        exitm
        endif
        endm
        endm

extd    macro   a,b,c,d,e,f,g,h,i,j,k
        irp     xx,<a,b,c,d,e,f,g,h,i,j,k>
        ifnb    <xx>
        extrn   xx:dword
        else
        exitm
        endif
        endm
        endm

extn    macro   a,b,c,d,e,f,g,h,i,j,k
        irp     xx,<a,b,c,d,e,f,g,h,i,j,k>
        ifnb    <xx>
        extrn   xx:near
        else
        exitm
        endif
        endm
        endm

extw    macro   a,b,c,d,e,f,g,h,i,j,k
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
        irp     xx,<a,b,c,d,e,f,g,h,i,j,k>
        ifnb    <xx>
        extrn   xx:word
        else
        exitm
        endif
        endm
        endm

pushm   macro   a,b,c,d,e,f,g,h,i,j,k
        irp     xx,<a,b,c,d,e,f,g,h,i,j,k>
        ifnb    <xx>
        push    xx
        else
        exitm
        endif
        endm
        endm

popm    macro   a,b,c,d,e,f,g,h,i,j,k
        irp     xx,<a,b,c,d,e,f,g,h,i,j,k>
        ifnb    <xx>
        pop     xx
        else
        exitm
        endif
        endm
        endm

        ENDIF           ; end MASM/TASM macros

        .list           ; end of asm.inc
```

LISTING 3

```
        title   order local labels
        include asm.inc
        .stack

;(C) Soft Advances 1989, All Rights Reserved

PROC_LINE_MAX           equ     100
DOUBLE_CASE_BITS        equ     2020h

        .data
        extb    err_file_too_big
        extw    error_text,dgroup_segment

write_output_flag       db      0       ; set this flag if input file changes

proc_text               db      'proc'
endp_text               db      'endp'
bak_text                db      'bak'

resequenced_text        db      ' resequenced',13,10,0
no_update_text          db      'sequence OK, nothing written',0

err_endp_missing        db      'ENDP missing',0
err_output_file         db      'Error creating, writing, or closing output',0
err_rename              db      'Rename failed',0
```



```
        .data?
        extd    argv
        extw    argc

label_count             dw      ?       ; number of labels in procedure body
label_table             dw      PROC_LINE_MAX dup(?)

output_bucket           dd      ?       ; output buffer pointer
output_bc               dw      ?

proc_initials           db      ?,?,?,? ; first 3 characters of local label

bak_filename            db      FILENAME_MAX dup(?)

        .code
 extn isalpha_,isdigit,isalpha,skip_spaces_tabs,isalnum,startup
 extn strskp_white,skip_line,ms_dos,read_entire_file,exit_program
 extn malloc_raw,outchr,outchr_asciiz_dgroup,fatal_error_handler

;;      atoi skip
;
;       entry   DS:SI   ascii digits
;       exit    AX      binary value
;               SI      updated to digit string delimiter
;               Cf      if bad number
;
atoi_update proc
        call    skip_spaces_tabs
        call    isdigit
        jne     ati4                    ; if bad or negative number

ati1:   push    dx                      ; process first digit
        sub     al,'0'
        cbw
        mov     dx,ax
        inc     si

ati2:   mov     al,[si]                 ; process each digit
        call    isdigit
        jne     ati3                    ;  if end of digits

        add     dx,dx                   ;  multiply by ten
        mov     ax,dx
        add     dx,dx
        add     dx,dx
        add     dx,ax

        lodsb                           ;  sum new digit
        sub     al,'0'
        cbw
        add     dx,ax
        jmp     ati2

ati3:   clc
        mov     ax,dx
        pop     dx
        ret

ati4:   cmp     al,'-'                  ; here for possible negative number
        jne     ati5                    ;  if bad number

        inc     si                      ;  else probable negative number
```

Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Otken code, continued

5

```
        mov     al,[si]
        call    isdigit
        je      ati5                    ;  if bad number

        call    ati1                    ; parse digits
        neg     ax
        clc
        ret

ati5:   stc
        ret
atoi_update endp

;;      atoi special
;
;       entry   DS:SI   ascii digits
;       exit    AX      binary value
;               SI      updated to digit string delimiter
;               Cf      if bad number or leading zeros
;
atoi_special proc
        cmp     bptr [si],'0'           ; check for leading zero in digit
        jne     atoi_update             ;  if no leading zero

        mov     al,[si+1]               ;  else leading 0, check for lone 0
        call    isdigit
        jne     atoi_update             ;  if lone zero, parse it
        stc
        ret
atoi_special endp

;;      check label sequence
;
;       exit    Cf      if labels out of sequence
;       uses    AX,BX
;
check_label_sequence proc
        mov     ax,0
        lea     bx,label_table[bp]

cl11:   cmp     ax,label_count[bp]
        je      cl12                    ; if all labels OK (Cf=0)
        inc     ax
        cmp     ax,ss:[bx]
        lea     bx,[bx+2]
        je      cl11                    ; if label correctly sequenced
        stc                             ; else indicate bad sequencing
cl12:   ret
check_label_sequence endp

;;      convert label index
;
;       entry   AX      old label index
;       exit    AX      new label index
;               Cf      if bad index
;
convert_label_index proc
        pushm   cx,di,es
        mov     es,dgroup_segment[bp]   ; search table for old label index
        lea     di,label_table[bp]
        mov     cx,label_count[bp]
        repne   scasw
```

6

```
        stc
        jne     cli1                    ;  if not found, return error

        mov     ax,di                   ;  else return correct index
        sub     ax,offset label_table
        shr     ax,1
        clc
cli1:   popm    es,di,cx
        ret
convert_label_index endp

;;      count proc lines
;
;       entry   DS:SI   procedure text
;               DX      end of proc offset
;       exit    CX      number of lines
;       uses    AX
;
count_proc_lines proc
        push    si
        mov     cx,0

cp11:   cmp     si,dx
        jae     cp12                    ; if end of procedure
        call    skip_line
        inc     cx
        jmp     cp11

cp12:   pop     si
        ret
count_proc_lines endp

;;      extract label number
;
;       entry   DS:SI   local label with colon delimiter
;       exit    AX      label #
;               Cf      if unsupported or nonlocal label
;
extract_label_number proc
        push    si
        lodsb
        call    isdigit
        je      eln1                    ; if OPTASM local label (1$:)
        cmp     al,proc_initials[bp]
        jne     eln3                    ; if bad label
        lodsb
        cmp     al,proc_initials[bp+1]
        jne     eln3                    ; if bad label
        lodsb
        cmp     al,proc_initials[bp+2]
        jne     eln3                    ; if bad label

        call    atoi_special
        jc      eln3                    ; if bad label index
        jmp     eln2

eln1:   dec     si                      ; here for OPTASM labels (1$:)
        call    atoi_special
        jc      eln3

        cmp     bptr [si],'$'
        jne     eln3                    ;  if dollar sign suffix missing
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

7

```
        inc     si

eln2:   cmp     bptr [si],':'
        je      eln4                    ;  if near label colon OK (Cf=0)
eln3:   stc                             ;  else set error flag

eln4:   pop     si
        ret
extract_label_number endp

;;      find local labels
;
;       entry   DS:SI   procedure text
;               CX      nonzero line count
;       exit    Cf      if unable to process procedure
;       uses    AX,BX
;
find_local_labels proc
        pushm   cx,si
        mov     label_count[bp],0
        lea     bx,label_table[bp]

fll1:   mov     al,[si]                 ; process procedure line
        cmp     al,SPACE_CHAR
        jbe     fll2                    ;  if no label on this line
        cmp     al,';'
        je      fll2                    ;  if comment then not label

        call    extract_label_number
        jc      fll3                    ;  if unknown label format

        mov     ss:[bx],ax              ; record label index in label_table
        add     bx,2
        inc     label_count[bp]

fll2:   call    skip_line               ; advance to next line
        loop    fll1
        clc

fll3:   popm    si,cx
        ret
find_local_labels endp

;;      find proc endp
;
;       entry   DS:SI   text ptr
;               SS:BX   points to either "proc" or "endp"
;       exit    SI      points to line after proc or endp
;               DX      points to start of proc or endp line
;               Cf      if end of file
;       uses    AX
;
find_proc_endp proc
        jmp     fpe2
fpe1:   call    skip_line               ; advance to next source line

fpe2:   mov     al,[si]                 ; check start of line for proc name
        cmp     al,NULL_CHAR
        je      fpe4                    ;  if end of file
        call    isalpha_
        jne     fpe1                    ;  if not name (must start w/ letter)
```

8

```
        mov     dx,si                   ; skip to token following name
fpe3:   lodsb
        call    isalpha_
        je      fpe3
        dec     si
        call    strskp_white

        mov     ax,[si]                 ; check for PROC or ENDP tokens
        or      ax,DOUBLE_CASE_BITS
        cmp     ax,ss:[bx]
        jne     fpe1                    ;  if not PROC or ENDP
        mov     ax,[si+2]
        or      ax,DOUBLE_CASE_BITS
        cmp     ax,ss:[bx+2]
        jne     fpe1                    ;  if not PROC or ENDP
        mov     al,[si+4]
        call    isalpha_
        je      fpe1                    ;  if not PROC or ENDP
        call    skip_line

fpe4:   add     al,-1                   ; (set Cf if AL==0)
        cmc
        ret
find_proc_endp endp

;;      get proc initials
;
;       entry   DS:SI   procedure body
;               CX      line count
;       exit    Cf      if unsupported label syntax
;       uses    AX
;
get_proc_initials proc
        pushm   cx,si
gpi1:   mov     al,[si]                 ; search for "3 initial" local label
        call    isalpha
        je      gpi2                    ;  if found (maybe)
        call    skip_line
        loop    gpi1
        clc                             ; no 3 initial label found (not error)
        jmp     gpi4

gpi2:   mov     proc_initials[bp],al    ; save initials 1, 2, and 3

        mov     al,[si+1]
        call    isalpha
        jne     gpi3                    ;  if nonalpha, is not local label
        mov     proc_initials[bp+1],al

        mov     al,[si+2]
        call    isalpha
        jne     gpi3                    ;  if nonalpha, is not local label
        mov     proc_initials[bp+2],al

        mov     al,[si+3]
        call    isdigit
        je      gpi5                    ;  if good "3 initial" local label

gpi3:   stc
gpi4:   mov     wptr proc_initials[bp],0
        mov     bptr proc_initials[bp+2],0
```

Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Otken code, continued

9

```
gpi5:   popm    si,cx
        ret
get_proc_initials endp

;;      itoa decimal
;
;       entry   AX      number
;               ES:DI   destination
;       exit    DI      updated (points to delimiting 0)
;       uses    AX
;
itoa_decimal proc
        cmp     ax,0
        jge     ita1                    ; if positive
        mov     bptr es:[di],'-'        ; else "stosb" minus sign
        inc     di
        neg     ax

ita1:   pushm   cx,dx
        mov     cx,10
        mov     dx,-1                   ; push sentinal
ita2:   push    dx
        mov     dx,0
        div     cx                      ; divide digits
        cmp     ax,0
        jne     ita2
        inc     dx
ita3:   xchg    ax,dx                   ; write digits
        add     al,'0'-1
        stosb
        pop     dx
        inc     dx
        jnz     ita3                    ;  if not sentinal
        mov     es:[di],dl              ; write delimiting 0
        popm    dx,cx
        ret
itoa_decimal endp

;;      main
;
main    proc
        mov     ax,@data                ; compensate for MS-LINK bug
        mov     ss,ax
        mov     sp,offset stack

        mov     ax,sp                   ; initialize segments, argv, argc, etc
        call    startup

        cmp     argc[bp],1
        jbe     mai3                    ;  if no arguments - exit

        lds     si,argv[bp]
        mov     si,[si+2]

        call    read_entire_file
        jc      mai5                    ;  if file not found or too big
        call    malloc_output_bucket
        jc      mai5                    ;  if not enough memory

mai1:   mov     cx,si                   ; find start of next procedure
        lea     bx,proc_text
        call    find_proc_endp
```

10

```
        jc      mai2                    ;  if end of file

        xchg    cx,si                   ; output source up to procedure
        sub     cx,si
        rep     movsb

        mov     cx,si                   ; find end of procedure
        lea     bx,endp_text
        call    find_proc_endp
        jc      mai4                    ;  if endp missing error

        mov     si,cx                   ; count lines in procedure body
        call    count_proc_lines
        jcxz    mai1                    ;  if empty proc, ignore it
        cmp     cx,PROC_LINE_MAX
        ja      mai1                    ;  if procedure too long, ignore it

        call    get_proc_initials
        jc      mai1                    ;  if unsupported local label syntax

        call    find_local_labels
        jc      mai1                    ;  if unsupported label syntax

        call    check_label_sequence
        jnc     mai1                    ;  if label sequence OK

        call    resequence_proc_labels

        mov     write_output_flag[bp],-1
        call    outchr_proc_name
        jmp     mai1

mai2:   xchg    cx,si                   ; output source up to procedure
        sub     cx,si
        rep     movsb
        sub     di,wptr output_bucket[bp] ; save output byte count
        mov     output_bc[bp],di

        call    update_asm_source

mai3:   jmp     exit_program

mai4:   mov     error_text[bp],offset err_endp_missing
mai5:   jmp     fatal_error_handler
main    endp

;;      malloc output bucket
;
;       entry   CX      file byte count
;       exit    ES:DI   output bucket
;               Cf      if file too big or not enough memory
;       uses    AX,CX
;
malloc_output_bucket proc
        add     cx,1024                 ; allocate oversized output bucket
        jc      mob2                    ;  if file too big
        call    malloc_raw
        jc      mob1                    ;  if not enough memory

        mov     wptr output_bucket[bp],di
        mov     wptr output_bucket[bp+2],es
mob1:   ret
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
mob2:   mov     error_text[bp],offset err_file_too_big
        ret
malloc_output_bucket endp

;;      outchr proc name
;
;       entry   DS:SI   points to endp line
;       uses    AX
;
outchr_proc_name proc
        push    si                      ; display procedure name
        mov     al,SPACE_CHAR
opn1:   call    outchr
        lodsb
        cmp     al,SPACE_CHAR
        ja      opn1
        pop     si

        lea     ax,resequenced_text     ; * resequenced*
        call    outchr_asciiz_dgroup
        ret
outchr_proc_name endp

;;      rename
;
;       entry   DS:SI   old name
;               ES:DI   new name
;       exit    Cf      if error
;               AX      dos error code
;
rename proc
        mov     ah,56h
        xchg    dx,si
        call    ms_dos
        xchg    dx,si
        ret
rename endp

;;      resequence local label
;
;       entry   A       first character of label
;               DS:SI   points to possible local label
;               ES:DI   output pointer
;       exit    DI,SI   updated
;       uses    AX,BX
;
resequence_local_label proc
        mov     bx,si
        cmp     al,'9'
        jbe     rl11                    ; if possible OPTASM label

        cmp     al,proc_initials[bp]    ; check for 3 initial label
        jne     rl12                    ;  if not
        mov     al,[si+1]
        cmp     al,proc_initials[bp+1]
        jne     rl12
        mov     al,[si+2]
        cmp     al,proc_initials[bp+2]
        jne     rl12

        lea     si,[si+3]               ; try parsing label index
```



```
        call    atoi_special
        jc      rl12                    ;  if no index, not label
        call    convert_label_index
        jc      rl12                    ;  if unknown label index

        xchg    bx,si                   ; copy procedure initials
        movsw
        movsb
        xchg    bx,si

        call    itoa_decimal            ; and write new label index
        ret

rl11:   call    atoi_special            ; here for possible OPTASM label
        jc      rl12                    ;  if bad index
        cmp     bptr [si],'$'
        jne     rl12                    ;  if dollar suffix missing
        call    convert_label_index
        jc      rl12                    ;  if unknown index
        call    itoa_decimal

        movsb                           ; copy dollar suffix
        ret

rl12:   mov     si,bx
        ret
resequence_local_label endp

;;      resequence proc labels
;
;       entry   DS:SI   procedure body
;               ES:DI   output pointer
;               CX      line count
;       exit    DI,SI   updated
;       uses    AX,BX,CX
;
resequence_proc_labels proc
rpl1:   mov     al,[si]
        call    isalnum
        jne     rpl3                    ; if not label
        call    resequence_local_label  ; else resequence it
        jmp     rpl3

rpl2:   movsb                           ; skip to next delimiter
rpl3:   mov     al,[si]
        cmp     al,SPACE_CHAR
        je      rpl4
        cmp     al,TAB_CHAR
        je      rpl4
        cmp     al,EOL_CHAR
        je      rpl5                    ;  if end of line
        cmp     al,';'
        je      rpl5                    ;  if comment
        cmp     al,','
        jne     rpl2                    ;  if not delimiter

rpl4:   movsb                           ; delimiter found, skip white space
        mov     al,[si]
        cmp     al,SPACE_CHAR
        je      rpl4                    ;  if white space
        cmp     al,TAB_CHAR
        je      rpl4                    ;  if white space
```

Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Otken code, continued

13

```
        cmp     al,';'
        je      rp15                    ;  if comment
        cmp     al,EOL_CHAR
        je      rp15                    ;  if end of line

        jmp     rp11                    ; end of white space, check for label

rp15:   lodsb                           ; skip to next line
        stosb
        cmp     al,EOL_CHAR
        jne     rp15

rp16:   loop    rp11                    ; loop while more lines left in body
        ret
resequence_proc_labels endp

;;      strcpy new ext
;
;       entry   DS:SI   asciiz source filename
;               ES:DI   destination pointer
;               SS:BX   new extension
;       exit    DI,SI   updated
;       uses    AX
;
strcpy_new_ext proc
        jmp     sne2
sne1:   stosb                           ; copy drive, directories, filename
sne2:   lodsb
        cmp     al,NULL_CHAR
        je      sne3                    ;  if end of filename
        cmp     al,'.'
        jne     sne1                    ;  if not start of extension

        mov     al,[si]
        cmp     al,'.'
        je      sne1                    ;  if ..
        cmp     al,'\'
        je      sne1                    ;  if .\

sne3:   mov     al,'.'                  ; write new extension
        stosb
        mov     ax,ss:[bx]
        stosw
        mov     al,ss:[bx+2]
        mov     ah,NULL_CHAR
        stosw
        ret
strcpy_new_ext endp

;;      unlink
;
;       entry   DS:SI   filename to delete
;       exit    Cf      if error
;               AX      dos error text
;
unlink  proc
        mov     ah,41h                  ; delete file
        xchg    dx,si                   ;  (unlink is ANSI C terminology)
        call    ms_dos
        xchg    dx,si
        ret
unlink  endp
```

14

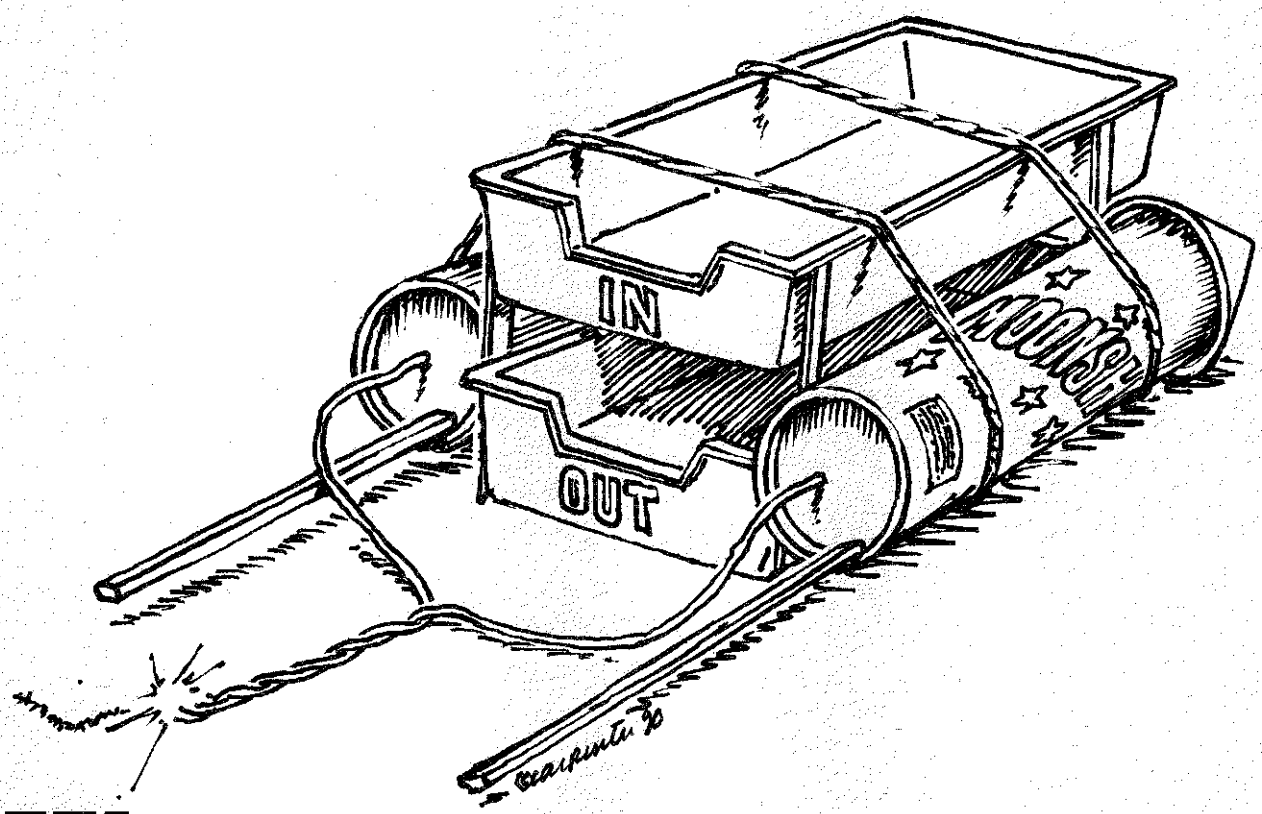
```
;;      update asm source
;
;       uses    AX,BX,CX,DX,DI,SI,DS,ES
;
update_asm_source proc
        test    write_output_flag[bp],-1
        jz      uas3                    ;  if nothing to update

        lds     si,argv[bp]             ; make .bak filename
        mov     si,[si+2]
        lea     bx,bak_text[bp]
        mov     es,dgroup_segment[bp]
        lea     di,bak_filename[bp]
        call    strcpy_new_ext

        mov     ds,dgroup_segment[bp]   ; delete old .bak file
        lea     si,bak_filename[bp]
        call    unlink                  ;  (ignore errors)

        lds     si,argv[bp]             ; rename .asm to .bak
        mov     si,[si+2]
        lea     di,bak_filename[bp]
        call    rename
        jc      uas1                    ;  if rename failed

        mov     ah,3Ch                  ; create .asm file
        mov     cx,0
        mov     dx,si
        call    ms_dos
        jc      uas2
        mov     bx,ax

        mov     ah,40h                  ; write .asm file
        mov     cx,output_bc[bp]
        lds     dx,output_bucket[bp]
        call    ms_dos
        jc      uas2

        mov     ah,3Eh                  ; close .asm file
        call    ms_dos
        jc      uas2
        ret

uas1:   mov     error_text[bp],offset err_rename
        jmp     fatal_error_handler

uas2:   mov     error_text[bp],offset err_output_file
        jmp     fatal_error_handler

uas3:   lea     ax,no_update_text[bp]
        jmp     outchr_asciiz_dgroup
update_asm_source endp

        end     main
```

[NOTE: The miscellaneous routines, which appeared in previous issues, are not printed here. They are available through our bulletin board services and on the code disk for this issue (see pages 8 and 93). —Eds.]

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Fast File I/O in QuickBASIC

by
Ethan Winer



Without doubt, one of the slowest operations a program performs is reading and writing data files to and from disks. In many cases, the speed (or sluggishness) is dictated by the physical-access delay of the disk device and the time required to transfer data based on the disk's rotation speed.

One exception, however, is when many reads or writes must be performed on small pieces of data. For example, it is quite common to save or load an entire numeric array. In the case of an integer array that contains, say, 10,000 elements, 10,000 individual calls to the DOS file I/O services will be needed. Even though DOS is written in assembly language, it still takes a fair amount of time to process each read or write request.

One obvious solution is to process the file operation as a single, large read or write; indeed, I have written assembly language routines to do just that in my company's QuickPak Professional add-on library product. It is also possible to call QuickBASIC's internal PUT and GET routines directly. By bypassing the QuickBASIC compiler and its syntax checking, you can coerce it into reading and writing up to 64K of data in a single operation. Larger files can be processed in pieces. The

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

trick is to determine the names of the routines and the number and type of parameters they expect to receive.

QuickBASIC versions 4.0 and later contain four different internal routines for reading and writing binary data. Two of these are meant for reading data from a file, with one using the current DOS SEEK location and the other accepting a long-integer SEEK argument. The other two routines are for writing data to disk.

Most of QuickBASIC's internal routines begin with the characters "B$", which are illegal in a subroutine name. Fortunately, the ALIAS keyword allows you to declare a procedure with two different names—the name you will use when calling it from the program, and the actual name that is made public for the linker. When Microsoft introduced interlanguage calling capabilities in QuickBASIC version 4.0, it needed a way to allow access to routines written in C. These routines always start with an underscore character, which is also illegal in a QuickBASIC procedure name.

The example program shown in Listing 1 declares the four internal routines as follows: **BigSave** writes data using the current DOS file-pointer position, and **BigSaveS** expects a SEEK argument. Likewise, **BigLoad** reads from the current file position, and **BigLoadS** requires an offset to which to SEEK before reading. All four of these routines require the parameters to be passed "by value" as opposed to "by address," which is BASIC's usual method of passing parameters. This results in code that is both faster and smaller, because an extra level of indirection is avoided; that is, the routines can obtain the values directly from the stack, rather than having to first determine an address and then go to that address for the actual value. Even though BYVAL and SEG *look* like they would result in additional code being added to a program, they are really just directives to the compiler.

Before any of these routines may be called, you must open the file to be read or written for binary operation. Then, the first parameter that each routine expects is the BASIC file number that was used to open the file. The address parameter is passed as a SEG value, which means that both a segment and offset are required. Notice that a file may be loaded to or saved from any area of memory by replacing [SEG Address] with [BYVAL Segment, BYVAL Address]. When SEG is used as part of a CALL statement, the "value" of the variable's segment is pushed on the stack, followed by the value of its address. Substituting two separate arguments "by value" is functionally the same thing as far as the routines are concerned. Also, notice that the internal routine names are not available within the QuickBASIC editing environment; therefore, this example program must be compiled to disk before it may be tested.

In my own informal tests, I have found this technique to be as much as ten times faster than reading or writing individual array elements using a BASIC FOR/NEXT loop. The actual savings will of course depend on the number of elements being processed and their length in bytes. Unfortunately, this method cannot be used with QuickBASIC string arrays, because they are not kept in consecutive memory locations. However, numeric arrays may be accommodated, as well as any fixed-length or user-defined TYPE array.

It is important to understand that when manipulating a fixed-length string array, you must not use the SEG operator. Whenever a fixed-length string or array element is used as an argument to an external routine, QuickBASIC first makes a copy of it into a regular string variable. Then, the address of the copy is passed instead. Since the address of a copy of an array element has no relevance to the address of the actual array, we must use a different approach.

In fact, there are two possible solutions. One is to create a TYPE definition that is comprised solely of a fixed-length string portion. Although the following example assumes a string length of 20 characters, you would of course use whatever is appropriate for your program:

Microsoft BASIC 7.0 Compiler

Microsoft has just announced a new version of their high-end BASIC compiler called the BASIC Professional Development System (PDS). Priced at $495, this product is intended for use by professional developers. Many limitations of the earlier QuickBASIC 4 and BASIC 6.0 compilers have been eliminated, and a large number of important new features have been added. Perhaps most exciting are the inclusion of an ISAM file handler and an expanded development environment that uses EMS memory to accommodate programs of nearly any size.

Other important additions include full support for overlays, string-data capacity greater than 64K, and a new Currency data type. This is a unique 8-byte "very long integer" that provides absolute accuracy in calculations, but without the performance degradation associated with BCD methods. The speed of regular floating point operations has also been improved—in some cases by as much as three hundred percent.

Because BASIC is often used for business and financial programming, Microsoft has bundled in several features from their Excel spreadsheet. For example, functions are provided for calculating internal rate of return, net present value, and future value. Other "toolbox" features include date and time arithmetic, presentation-quality graphics, and routines for sophisticated numeric formatting.

BASIC has long been the language of choice for many programmers, and with the introduction of version 7 it is clear that Microsoft will continue to expand and improve the language. Although they may meet some resistance to the $495 price tag, Microsoft has gone to a great deal of effort to provide as much value as possible. From my perspective as a professional BASIC programmer, this is a truly impressive product.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
TYPE FLen
   S AS STRING * 20
END TYPE
DIM Array(1 TO 10000) AS FLen
```

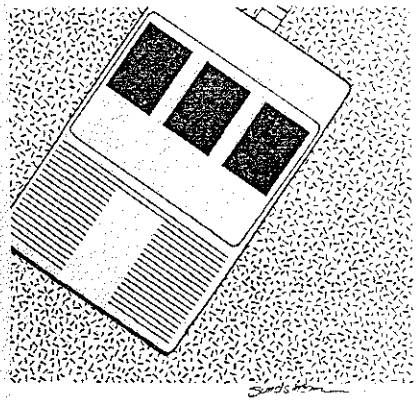
The second solution is to use a combination of BYVAL VARSEG and BYVAL VARPTR to pass the segment and address of the starting array element directly. When QuickBASIC sees VARSEG or VARPTR, it realizes that you do in fact want the actual segment and address of the specified array element. Thus, you would use the following syntax when calling BigSave to save a fixed-length string array:

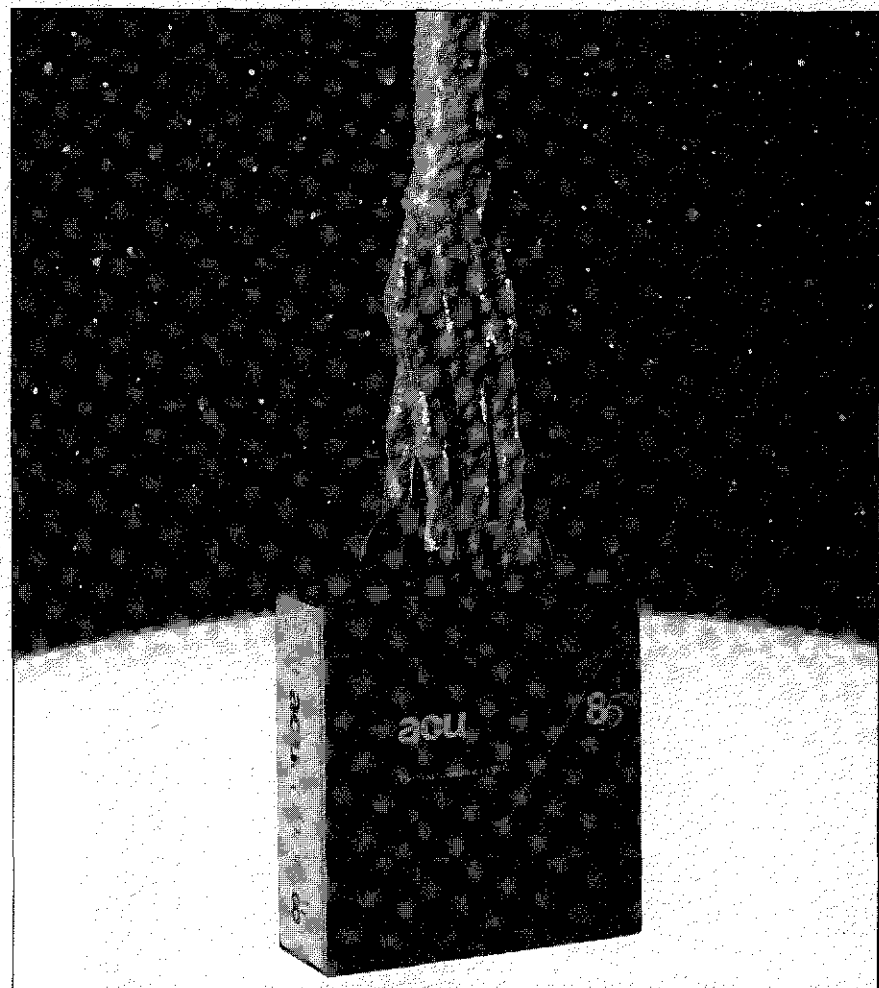
```
CALL BigSave(FileNumber,
   BYVAL VARSEG(Array(First)),
   BYVAL _ VARPTR(Array(First)),
   NumBytes)
```

One final note concerns saving or loading more than 32,767 bytes. QuickBASIC does not support unsigned integers, and so you must use an equivalent negative value. This is quite easy to determine: Simply subtract 65,536 from the required number of bytes. It is a common trick to avoid negative numbers when calling assembly language routines by substituting a long-integer number or variable. However, that will not work in this case, because two extra bytes will be pushed onto the stack by the use of BYVAL. Therefore, it is essential that you specify the correct types of parameters when calling these routines.

Ethan Winer is president of Crescent Software and the author of QuickPak Professional and P.D.Q. He may be reached at 11 Grandview Avenue, Stamford, CT 06905; by phone at 203-846-2500; on CompuServe as 72657,3070; and on MCI Mail as EWINER.

Code follows.





INTRODUCING
THE MOST PORTABLE COBOL
IN THE UNIVERSE

Go where you want to go
Transport COBOL programs to almost any platform—without changing one line of code. Run on AIX, MS-DOS, NETBIOS, UNIX, OS/2, ULTRIX, XENIX and VMS without recompiling. No other COBOL-85 offers such hardware independence, or the instantly-expanded possibilities.

Open new windows in your applications
Create better-looking programs with windows. Add help screens, pop-up/pull-down menus, context-sensitive messages, more. ACUCOBOL-85 has a Display Window verb, including extensions for boxed, reverse-video and pop-up windows in color. Paint special forms on the screen with our Display Line verb. And scroll up/down inside windows.

Don't get lost in space:
ACUCOBOL-85 keeps you on course
ACUCOBOL-85 is NBS certifiable error-free under the High 1974 and Intermediate 1985 ANSI standards. If you lose course, it has its own interactive source code debugger that's fast and easy to use.

Step into the future
Make your job simpler and stay competitive—even ahead of the game. Put the next generation COBOL to work for you today.

Free evaluation copy
Call (619) 271-7097 today for your free evaluation copy of ACUCOBOL-85. And see the shape of things to come.

acucobol inc

Voice: (619) 271-7097 Fax: (619) 566-3071
7950 Silverton Avenue Suite 201 San Diego California 92126

ACUCOBOL-85/the ACUCOBOL logo are trademarks of ACUCOBOL, Incorporated; MS-DOS/XENIX of Microsoft Corporation; NETBIOS/OS/2/AIX of International Business Machines Corporation; VMS/VAX/ULTRIX of Digital Equipment Corporation; UNIX of AT&T Bell Laboratories; RM/COBOL and RM/COBOL-85 of Ryan-McFarland Corporation.

Number 282 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

All code in this issue is available through our bulletin board services or directly from *PJ* on disk. See pages 8, 88, 22, and 93.

Code from "Fast File I/O . . ." by Ethan Winer

```
LISTING 1

'********** BIGPUT.BAS - demonstrates saving and loading an entire array

'Copyright (c) 1989 Ethan Winer

DEFINT A-Z

DECLARE SUB BigSave ALIAS "B$PUT3" (BYVAL FileNumber, SEG Address, _
    BYVAL NumBytes)
DECLARE SUB BigLoad ALIAS "B$GET3" (BYVAL FileNumber, SEG Address, _
    BYVAL NumBytes)
DECLARE SUB BigSaveS ALIAS "B$PUT4" (BYVAL FileNumber, BYVAL SeekLocs, _
    SEG Address, BYVAL NumBytes)
DECLARE SUB BigLoadS ALIAS "B$GET4" (BYVAL FileNumber, BYVAL SeekLocs, _
    SEG Address, BYVAL NumBytes)

CONST NumEls% = 10000                   'the size of the test array
DIM Array(1 TO NumEls%)                 'create an integer array
FOR X = 1 TO NumEls%                    'initialize it to ascending values
   Array(X) = X
NEXT

OPEN "BigTest" FOR BINARY AS #1         'create the output file
BigSave 1, Array(1), NumEls% * 2        'file #1, array start, number of bytes
CLOSE #1                                'close the file

FOR X = 1 TO NumEls%                    'clear the array to prove we loaded
   Array(X) = 0                         '  it again later
NEXT

OPEN "BigTest" FOR BINARY AS #1         'open the file again
BigLoad 1, Array(1), NumEls% * 2        'load the entire array at once
CLOSE #1                                'close the file

FOR X = 1 TO NumEls%                    'print the results to prove it works
   PRINT Array(X);
NEXT
```


MISOSYS

Programmer's Journal downloads are now available from MISOSYS' forum on Compuserve: GO PCS49

Are you still fussing with floppies for BACKUP? Our QIC40 JUMBO tape drive from MISOSYS is your solution!

The Colorado Memory Systems' JUMBO tape drive is one drive that fits all computers. It comes ready for internal use in AT's, XT's, and PC's, and connects to your floppy disk controller. Our adaptor board can be used to connect JUMBO when all FDC connectors are in use. Kits are available for XT/AT or PS/2 machines which convert Jumbo to external use.

- In about 5.5 minutes, JUMBO backs up 10MB's file-by-file - the fastest in the industry! 40MB's gets backed up in about 18 to 20 minutes. Uses industry-standard DC2000 tape cartridges.
- JUMBO plugs into your floppy disk controller to save cost, power, and a slot. Needs 5-1/4" (or 3.5" with faceplate) mounting slot.
- Our tape adapter board mounts in your host computer to provide an additional tape port — especially useful in 286 and 386 computers. When used with our external JUMBO, it lets you share your drive between computers. Note: external adaptor includes "Tape Adaptor"
- JUMBO has custom chips, high speed brushless motors, automatic circuits, no pots to vibrate out of calibration, and fewest parts to assure long life and technical leadership for years to come.

DJ10 Jumbo	$295 ($5S&H)
Tape Adaptor	$75 ($3S&H)
External Adaptor	$110 ($5S&H)
DC2000 tape	$22.50

M/C & VISA. S&H are U.S. only.

MISOSYS, Inc.
P.O. Box 239
Sterling, VA 22170-0239
800-MISOSYS
or 703-450-4181

Now get the clearest view of your disk drive and memory contents with our DED86™

Powerful features in Version 2!

When you need to travel through your disk drive, why settle for a tool that isolates you from the road? DED86 gives you the direct controls you need to explore your disk. It's a full-screen sector-oriented disk/file editor and a page-oriented memory editor. When you want to "unerase" erased files, DED86's flexible KEEP facility does the job without you fussing over FATs.

Check out this list of features

- ✓ Look by cylinder/head/sector, sector or cluster; even reserved/hidden sectors
- ✓ Scan free clusters to search for erased data ✓ Jump about subdirectories
- ✓ Keep sectors & clusters for writing to a file; Great for partial file recovery
- ✓ Edit bytes in hexadecimal or ASCII, zap in 0s; Undo last edit!
- ✓ Search your disk drive, a file, or memory for ASCII or hexadecimal strings
- ✓ Touch a directory file entry with your date and/or time
- ✓ Obtain complete disk statistics in one screen; includes usage data
- ✓ Alter file attributes: archive, system, hidden, read
- ✓ Save/Restore sectors to/from auxiliary buffers; for moving data around
- ✓ DOS subshell available while using DED86 lets you shell to other programs
- ✓ Handles 5.25" & 3.5" drives & RAM disks; even large partition drives

DED86 reduced by more than 50%; Now only $29.95 + $2S&H

Fujitsu Floppy Disk Drives:	5.25" 360K 1/2 height	$75+$4S&H
	3.5" 720K in 5.25" frame	$85+$3S&H
	3.5" 1.44M in 5.25" frame	$99+$3S&H

Number **121** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

AtLast! Announces Two New Products

AtLast! Software has announced two new products for DOS—Overlay Architect and Overlay Optimizer. These products are designed to be used in conjunction with overlay linkers (Plink86plus or .RTLink) to create nested overlays automatically. Overlay Architect scans object and library modules to determine calling relationships. The generous limits are 16,000 modules on a 640K PC. Overlay Optimizer increases performance of overlaid programs by analyzing the overlay structure to create better-optimized nested overlays. Overlay Architect costs $369, and Overlay Optimizer is $269; they can be purchased together for $569. AtLast! Software also offers a service to build custom overlay structures for developers.

AtLast! Software
449 Mountain View Road
Boulder, CO 80382
303-938-1210

Borland Releases Paradox Engine

Borland has announced the Paradox Engine, a C library that opens the architecture of their Paradox relational database to developers. The Paradox engine enables C programmers to build applications that create or access Paradox data. The API consists of more than 70 functions to manipulate Paradox tables in single and multi-user environments. The Paradox Engine retails for $495, with a special introductory offer of $195 for registered Borland users. A Pascal version is planned for later in 1990.

Borland International
P.O. Box 660001
Scotts Valley, CA 95066
408-438-8400

CLEAR+ for C Generates Charts

CLEAR Software has recently released version 1.08 of CLEAR+ for C, a follow-up to their successful earlier product, CLEAR+ for dBASE. Taking C source code as input, CLEAR+ for C can generate program flow charts, hierarchical (function-calling) tree charts, function cross-references, function-prototype files, and formatted source listings. The CLEAR acronym stands for Clarify Logic for Enhancement, Analysis, and Review. Clearly (no pun intended), the flow charts and tree diagrams are the real meat of this product. The generated charts can be viewed on a graphics screen or output to a printer. The output can also be exported to PCX (PC Paintbrush) or PIC (PC Paint) files. CLEAR+ for C costs $199.95, the same price as the dBASE version.

CLEAR Software, Inc.
1440 Beacon Street
Brookline, MA 02146
800-339-1758
617-232-4720

Crescent's P.D.Q. Lives Up to Its Name

Crescent Software has released P.D.Q.—a powerful library for use with QuickBASIC. P.D.Q. is a replacement linking library for use with Microsoft QuickBASIC 4.0 or later. The primary purpose of P.D.Q. is to create programs that are as small and fast as possible. When a compiled BASIC program is linked with PDQ.LIB instead of the usual BCOM library supplied with QuickBASIC, the EXE file size will be reduced dramatically and the execution speed will be increased. The P.D.Q. library is written in assembly language to optimize execution speed. P.D.Q. costs $99.

Crescent Software
11 Grandview Avenue
Stamford, CT 06905
203-846-2500

ProtoFinish for Prototyping and Demos

Genesis Data Systems has released ProtoFinish, a tool for developing prototypes, demos, presentations, and tutorials. ProtoFinish is a major upgrade from their earlier Rapid Application Development System (RADS). The ProtoFinish System allows you to design text screens quickly and easily, capture screens from existing applications, create sounds and music, and combine these parts into a prototype or demo using a powerful prototyping language. ProtoFinish also includes a library of routines and associated utilities to generate source code to display screens, play music, and so on, from these prototypes. Languages supported include Microsoft Business and QuickBASIC, Turbo Pascal, Turbo C and Microsoft QuickC, Clipper (dBASE language), and assembly language. Utilities also support capturing CGA graphics screens and various screen-conversion utilities. ProtoFinish image data can be converted to assembly language data, BASIC data statements, arrays in Pascal or C, ASCII data, and COM files. A royalty-free run-time program is included for distributing prototypes and demos. Technical support is available via a toll-free 800 number. ProtoFinish is priced at $299.95.

Genesis Data Systems
8415 Washington Place NE
Albuquerque, NM 87113
800-777-1437
505-821-9695

X Window Servers for DOS Updated by GSS

Graphics Software Systems (GSS) has released version 1.3 of PC-Xview/16 and PC-Xview, X Window servers for DOS. These products allow a DOS AT-compatible with either an EGA, VGA, or video card supporting DGIS (many TI 34010 boards) to function as an X Window server when connected to a host machine. PC-Xview requires an 80286/80386 with 640K, while PC-Xview/16 can use extended memory for better memory management. PC-Xview/16 comes with a version bundled with Rational Systems' DOS extender, which allows use of memory above 1 megabyte by the X Window server. Network connections are now supported using *LanWorkplace* for PC DOS, PC/TCP (from FTP Software), and PC-NFS. PC-Xview/16 costs $395, and regular PC-Xview is $295. Upgrades from earlier versions to PC-Xview/16 are $100.

Graphics Software Systems, Inc.
9590 S.W. Gemini Drive
Beaverton, OR 97005
503-641-2200

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Graphics User Interface Toolkit for DOS

Ithaca Street Software has released Menuet, a graphics user-interface toolkit for DOS. The current version is designed to run on top of MetaWINDOW (from Metagraphics). Since MetaWINDOW supports a very large list of video graphics cards, from Hercules to high-end TI 34010, Menuet provides similar extensive device support. MetaWINDOW provides 200 graphics functions, and Menuet adds 400 additional functions, although a small subset can be used to create a Windows-style environment. Menuet provides movable, resizable windows, pull-down and bar menus, dialog boxes, data-entry forms, scrollable lists, checkboxes and radio buttons, icons and icon bars, on so on. Screens are saved and restored to EMS memory if available. PCX, GEM image, First Publisher ART (clip art), and Microsoft Paint image files can be read into MetaWINDOW bit maps. Menuet also works with Ithaca's Icon Tools/Plus, utilities and functions to draw icons. Menuet is supplied as a C-compatible library (Turbo C, Microsoft C, and Zortech C++ versions are available) to be linked with your application. An Interface Design Tool (IDT) is also included for prototyping application shells and generating C source code for Menuet-based programs. The generated source code includes stubs for the developer to complete the application. Menuet costs $325, and the source code is available for an additional fee.

Ithaca Street Software, Inc.
1145 Ithaca Drive
Boulder, CO 80303
303-494-8865

JPI Releases DOS and OS/2 ANSI C Compilers

Jensen and Partners (JPI) has released TopSpeed C, new ANSI-conforming C compilers and integrated development environments. Two DOS versions as well as an OS/2 version are available. All versions of the Standard Edition include an optimizing compiler supporting a full implementation of ANSI C, ANSI standard libraries, extended libraries to make full use of DOS, a make utility, a high-speed linker, a librarian, a multiwindow source-level debugger, and integrated environment. Both command-line and integrated-environment versions of the compiler and utilities are included. The environment includes a multiwindow/multifile editor.

The Extended Edition adds full source-code (ASM and C) to libraries, Microsoft Windows support, DOS Dynamic Linking (DLL), an assembler, a disassembler, an execution profiler, and a post-mortem debugger. The compiler executes very fast and generates highly optimized, compact code. TopSpeed C defaults to passing parameters in registers (like Watcom C), but pragmas provide detailed control of such options. Functions can be defined and expanded inline for high-speed execution. A high-speed graphics library compatible with Microsoft C is supplied, along with an interface to the Borland BGI. A process module provides multithread execution within a single process under DOS. The linker is smart—it links only those functions from an object file that are required.

The OS/2 Edition includes all of the features of the DOS Extended Edition, with the compiler and utilities to run in the protected mode under OS/2. Both OS/2 and DOS programs can be created. The OS/2 Edition supports protected-mode code generation, automatic creation of DLLs, I/O privileged-mode functions, and a full interface to Presentation Manager (300K of include files). Full library source is included.

The Standard Edition is priced at $199, the Extended Edition is $395, and the OS/2 Edition sells for $495.

TechnoJock's Turbo Toolkit

Attention Turbo Pascal programmers.....

Cut your program development time in half and create sophisticated user interfaces in minutes with *TechnoJock's Turbo Toolkit!*

The *Toolkit* contains twelve units with all source code included, and a 300 page manual full of examples. These range from a comprehensive unit that provides a Sidekick Plus style directory window with all the expected sorting features, to a basic procedure that writes text centered on the screen.

And it's easy to use! If you've bought other toolkits but didn't use them because of all their complicated rules ... TTT is for you.

TECHNOJOCK SOFTWARE

- **ultra fast screen writing**
- **EGA/VGA extended lines**
- **exploding boxes**
- **virtual screens**
- **sorted directory listings**
- **comprehensive form input**
- **pulldown menus**
- **nested menus**
- **string manipulation**
- **windows**
- **full mouse support**
- **keyboard control**
- **date manipulation**

"TechnoJock's Turbo Toolkit may just cut your work in half! It provides easy-to-implement procedures.... designed for all Turbo Pascal programmers. With this Toolkit everyone becomes a TechnoJock."
Turbo Users Group

"Here, despite the whimsical name, is shareware at its best: an innovative product, a superb manual, quick mail-order service, and a reasonable price... If you're programming in Turbo Pascal, you absolutely must have TTT!"
Public-Domain Software & Shareware 2nd edition

"Looks great! We have decided to implement your Toolkit in in-house programs for Tech Support."
Rich Kubota
Technical Support
Borland International

TechnoJock Software, Inc.
PO Box 820927
Houston, TX 77282-0927

Orders: 1-800-242-4775
Enquiries: (713) 493-6354

Only $49.95!

Plus $5.00 shipping and handling (Canada $10.00, overseas $15.00). Texas residents add 8% Sales Tax.

Send checks to TechnoJock Software. Send Visa/Mastercard details to: Public (Software) Library PO Box 35705 Houston TX 77232-5705

Number 289 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Unleash Your 80386 & 80486!

If you own or are about to buy a 386, 386SX, or 486-based system, you are losing 50% or more of your system's speed if you are still running 16-bit code or have not installed a coprocessor. To date, hundreds of mainframe applications have been ported to the 386 that take advantage of the real power of the 386. Many of these new applications were compiled with MicroWay NDP Fortran and C. Our compilers break the 640K barrier wide open, making it possible to run programs up to 4 gigabytes in size. They also run in virtual memory under DOS, UNIX or XENIX, generating mainframe quality, globally optimized code capable of driving every coprocessor on the market, including the latest high performance devices from Weitek and Cyrix. If you have a question about coprocessor performance, call for a free copy of an article by Stephen Fried, "The State of PC Numerics in 1990".

For Model 70 and 80 owners, we offer a Weitek Micro Channel card that runs 200% faster than an 80387. For Compaq Deskpro owners, our Weitek daughterboard takes a 3167 and Cyrix CX83D87. We also offer RAMpak one and four megabyte Deskpro RAM upgrades.

NDP Fortran-386 is as close to VMS FORTRAN as you can get in a non-VAX environment, while our C conforms to both the UNIX System V and ANSI standards and includes MicroWay and Microsoft extensions. If you plan to mix FORTRAN with C or Pascal, our languages are the only choice. Our compilers include GREX, a library of 135 character and pixel oriented graphics routines that automatically detect and support CGA, Monochrome, Hercules, EGA and VGA. We complement our compilers with a complete line of 386 tools listed below.

486 Your PC!

Number Smasher-486™ is a 25 MHz replacement motherboard for ATs and 80386s. This motherboard supports an optional Weitek 4167 numeric coprocessor and up to 16 megabytes of 32-bit page mode memory. The Number Smasher-486 has been designed with Cache Shadow RAM for BIOS and video, and a 32-bit BURST memory bus which takes advantage of the enhanced features of the 486. It is ideal for applications such as CAD/CAE workstations, statistical analysis, linear programming, and multi-tasking operating environments, such as UNIX, XENIX, and OS/2. It supports the Phoenix, AMI, or Award BIOS. The Number Smasher-486 with 0K is priced at $3995.

NDPFortran-486™, NDPC-486™, and **NDP Pascal-486™** compilers will be released in March 1990 at $1195 each.

Please call (508) 746-7341 for more information.

386 Compilers and Tools

NDP Fortran-386™, NDP C-386™, and **NDP Pascal-386™** compilers generate globally optimized, mainframe quality code that runs on the 386 or 486 in protected mode under UNIX, XENIX, or Phar Lap extended DOS. The compilers address 4 gigabytes of memory while supporting the 80287, 80387, Weitek and Cyrix coprocessors. Applications can mix code from all three compilers and assembly language. The DOS versions allow the user to write his own numeric error handlers and interface 386 real mode programs from protected mode. The VM versions use Phar Lap's Virtual Memory Manager to run programs which exceed the size of your system memory. **NDP Fortran-386** is a full FORTRAN 77 with FORTRAN 66, BSD 4.2, DOD, and VMS extensions. **NDP C-386** is a full K&R C with both MS and ANSI extensions. It is 100% compatible with UNIX C and is substantially faster than the C which comes with UNIX. **NDP Pascal-386** is a full ANSI/IEEE Pascal, with extensions from C and BSD 4.2 Pascal.

DOS versions (require Phar Lap Tools)	$595
VM versions (requireVMM and Tools)	$695
UNIX/XENIX versions	$795
Phar Lap Development Tools	$495
Phar Lap Memory Manager (VMM)	$295
NDPWindows™ Library: $125, C Source:	$250
NDPHOOPS™	$575
NDP Plot™	$325
NDP/FFT™ NDP or 80x87 version ... ea.	$250
NDP to HALO '88 Graphics Interface	$100

NDP NAG™ — The NAG Workstation library is a subset of the NAG mainframe libraries. It contains a library of 268 routines designed to solve differential equations and eigenvalue problems, perform matrix operations, fit curves, do statistics and regression analysis, generate random numbers, and compute special functions and integrals: $795, Weitek version: $995

NEW! C++

NDP C++ is a MicroWay port of the UNIX C++ preprocessor version 1.2. It runs in protected mode under DOS, UNIX, or XENIX, and is ideal for writing numerics and graphics applications. The product comes with an example of how to support complex numbers in C++ $295

Parallel Processing

MicroWay's IBM compatible Monoputer, Quadputer, Videoputer, and Linkputer boards work together using Inmos transputers to provide expandable, plug-in mainframe performance for your desktop PC.

Monoputer™ — Includes one T800 and up to 16 meg of RAM for parallel code development. The 4 MWhetstones T800 makes it the ideal FORTRAN engine for cost-effective execution of your mainframe programs... from $1295

Quadputer™ — This board for the AT or 386 can be purchased with 1 to 4 transputers and 1 or 4 meg of memory per transputer. Two or more Quadputers can be linked together to build networks of up to 100 or more transputers providing mainframe power ..... from $1995

Videoputer™ — The highest performance graphics card on the market. Uses a T800 and TI 34010 in conjunction with an 80 MHz Brook-Tree DAC. ..... With one meg $4495

Linkputer™ — Links up to 8 boards to provide dynamic transputer topologies $1500

Transputer Compilers and Utilities
These parallel languages are designed for use with the Monoputer, Quadputer and Videoputer.

Logical Systems Parallel C	$595
3L Parallel C, FORTRAN, or Pascal	$895
TBUG — debugger for 3L	$330
Parsec Parallel C/static	CALL

ParaSoft EXPRESS Package — Includes transputer communications libraries, parallel code development library, C source level debugger, and system performance monitor ... $1500

Helios PC/s	$1250
Nexis Windows File Server — Lets you run parallel applications under the Microsoft Windows environment	$495
T800/NAG™ — Port of the complete NAG mainframe library. Contains 570 functions:	$2750

387BASIC™ — Our 16-bit MS compatible compiler introduces numeric register variables to produce the fastest 80x87 code on the market. For "floating-point and other complicated mathematical calculations, you'll appreciate the extraordinary speed with which 387BASIC handles these processes". PC Magazine 10/31/89 ..... $250

NEW! Cyrix FasMath™

Cyrix CX83D87 FasMath™ — Fastest 80-bit Intel compatible coprocessor. Performs transcendentals up to 3 times faster than the 80387.
20 MHz: $647 **25 MHz**: $814 **33 MHz**: $994

Weitek-Based Coprocessor Boards

mW1167™ and **mW3167™** coprocessor boards are built at MicroWay using Weitek components. Each includes an 80387 socket.

mW1167-16	$695
mW1167-20	$895
mW1167 Micro Channel-16/20	from $995
mW3167 Micro Channel-25/33	from $1795
3167-20	$995
3167-25	$1295
3167-33	$1695
mW3167/80387 Board	$200

Intel Coprocessors and RAM

8087	$84	**8087-2**	$120
80287-8	$195	**80287-10**	$220
80387-16	$330	**80387-16SX**	$310
80387-20	$375	**80387-25**	$460
80C287A	$280	**80387-33**	$550

287Turbo-20™ This coprocessor board runs a specially qualified Intel CMOS 80287 at 20 MHz regardless of the main CPU speed .. $450
RAMpak™ - one or four meg 32-bit memory expansion module for Compaq Deskpro 386 20/25. One meg .. $240, Four meg .. $700

386 Your AT!

NUMBER SMASHER-386™ — A full-sized card that replaces the 80286 microprocessor on your IBM AT or compatible motherboard with an 80386 that runs at 25 MHz. It runs numerically intensive applications up to a factor of 60 times faster, while maintaining full hardware and software compatibility when running all 386 applications. Includes sockets to optionally add up to 8 megabytes of 32-bit memory, an Intel 80387, Weitek, or Cyrix numeric coprocessor, and 64K or 256K of high speed cache memory..... from $1195

12 MHz PC Accelerators

SuperCACHE-286 12 MHz	$399
FastCACHE-286 12 MHz	$299

MicroWay

World Leader in PC Numerics

Corporate Headquarters: P.O. Box 79, Kingston, MA 02364 USA (508) 746-7341
32 High St., Kingston-Upon-Thames, U.K., 01-541-5466
USA FAX 508-746-4678 Italy 02-74.90.749 Holland 40 836455 Germany 069-75-2023

Number **104** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

JPI has also announced a February release date for version 2.0 of their Modula-2 and an April release date for their new Pascal product. All of these language environments are based on the same code generator. These appear to be impressive, *hot* new products for programmers.

Jensen and Partners International
1101 San Antonio Road, Suite 301
Mountain View, CA 94043
800-543-5202
415-967-3200

High C 1.6 for 386 & 486 DOS from Metaware

MetaWare has released Version 1.6 of High C Compiler for MS-DOS and 386DOS/486DOS. A new compiler option will optimize code for the 80486. Version 1.6 features expanded libraries, an editor, both a b-tree library and a graphics library for 386 DOS extenders, and a make utility. The 386 version will come bundled with the GFX/386 Graphics library from C Source, the Blackstar/386 C function library and BPTPlus from Sterling Castle, the EC editor from C Source, and Hyperdisk disk cache from HyperWare. The High C libraries are now ANSI conformant. The most attractive feature is that the MetaWare libraries have added functions compatible with Microsoft C. Maybe now High C developers won't have to write their own MSC-compatible library functions. High C for Extended DOS 386/486 costs $895, while High C for MS-DOS 80X86 is $595. Upgrades for current users are $150 for the 386 version and $100 for the 8088/80186/80286 version.

MetaWare Incorporated
2161 Delaware Avenue
Santa Cruz, CA 95060
408-429-6382

New C Tools from MMCAD Systems

MMC AD Systems has released version 2.0 of C Programmer's Toolbox/PC, a powerful set of tools for C programmers. The toolbox contains 21 tools in two volumes designed to enhance C-programmer productivity. The new revision-2.0 release now includes support for the ANSI C standard along with most common DOS C-compiler dialects. UNIX-style command-line arguments are also supported. In Volume I, major enhancements have been made to CFlow (program-function-flow diagrammer), CPrint (source-code reformatter), and Pmon (performance monitor). To assist with C++ and other object-oriented code, CFlow now optionally reports on function-pointer and function-address usage. In Volume II, major enhancements have been made to CLint (lint utility) and CXref (symbol cross-reference). Both tools now support ANSI C prototypes, recursive macro processing, new macro operators, some new target compilers, and greater options for reporting and type checking. An ANSI C– and K&R C–compatible preprocessor built into CFlow and CXref can also be used to expand macros and discard unused code. CFlow and CXref also incorporate a virtual-memory system so that programs of any size can now be analyzed. The other tools have also been improved. Volumes I and II sell for $99.95 each or $175 for both. Upgrades are priced at $30 per volume or $50 for both.

MMC AD Systems
Box 360845
Milpitas, CA 95035
408-263-0781

MKS Toolkit for OS/2 Based on UNIX Model

Mortice Kern Systems (MKS) has released the MKS Toolkit for OS/2, a large collection of utility programs modeled after their UNIX counterparts. MKS has also released version 3.1 of the MKS Toolkit for DOS. The MKS Toolkit contains more than 150 commands compatible with UNIX system V.3. These utilities can be linked together through pipes and I/O redirection to perform complex tasks. Among the tools included are the Korn Shell, AWK, VI, ed/sed, sort/merge, profiler, egrep/fgrep, spell, m4, mailx, compress, tar, and uuencode/uudecode. This latest version of the MKS Toolkit incorporates increased POSIX compatibility, extended regular expressions, and convenient menu-driven installation. The much-expanded documentation also covers file formats (cpio, tar, etc.) as well as shell- and editor-customization features. The OS/2 version costs $349, the DOS version is $249, and the two may be purchased together for $399. Upgrades are available from DOS to OS/2 and from older DOS versions.

Mortice Kern Systems Inc.
35 King Street North
Waterloo, Ontario, Canada N2J 2W9
519-884-2251

Opus Releases Revised MAKE Utility

OPUS has released version 5.2 of OPUS MAKE, their program-maintenance utility for DOS and OS/2. New enhancements include support for TLIB (Burton Systems), using time stamps in libraries created with Microsoft LIB and SLR Systems OPTLIB, regular expressions in macro modifiers, automatic chaining of inference rules, and choice of initialization file. Better controls are also available for associating properties with targets when using MAKE recursively. OPUS MAKE comes bundled with MKMF, a makefile dependency generator, which has also been enhanced to support regular expressions and flexible include-file processing.

Opus Software
1032 Irving Street, Suite 439
San Francisco, CA 84122
415-664-7901

FTN77/386 Version 2 Released by OTG

Salford Software Marketing has released version 2.0 of FTN77/386, a 386 FORTRAN compiler and development environment. Distributed by OTG Systems, version 2.0 now adds support for the Weitek 1167/3167 math coprocessors. The development environment includes the compiler with in-line 32-bit assembly language capabilities, linker, librarian, and debugger. The custom 386 DOS extender also supports virtual memory. Real-mode libraries can be called from a protected-mode program, and an interface to Btrieve is supplied. Version 2.0 costs $1,095.

OTG Systems, Inc.
P.O. Box 239
Clifford, PA 18413
717-222-9100

Dis.Doc Professional Version 3.1 Disassembler

RJSwantek has released version 3.1 of Dis.Doc Professional, a disassembler and patcher for the 80X86 family. Version 3.1 supports a batch mode for major changes and an interactive mode for quick minor edits during the disassembly process. Dis.Doc can disassemble any MS-DOS file or system-memory range. The batch mode uses up to eight passes for accuracy. All i86 processors and modes are supported from the 8086/87 through the 80486. Both 16-bit and 32-bit segments are supported for 80386/387 and 486 CPUs. The program supports using definition files and automatic labeling and commenting of interrupts, ports, driver interfaces, and BIOS data. The interactive mode allows searching for text strings, binary data, subroutines, code/data boundaries, and constants. The patch facility allows minor changes to be made to programs without assembling and linking. A utility program is also provided to unpack packed EXE files for disassembly. This

Continued on page 90

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C CODE FOR THE PC

source code, of course

	MS-DOS File Compatibility Package (read & write MS-DOS file systems on non-MS-DOS computers)	$750
	CSource Application Program Generator by MBAC (includes all source code; generator & libraries)	$500
	dB2c (dBase-to-C translator; includes db_Files for C and db_Tools for C)	$325
	CQL Query System (SQL retrievals on B-trees plus windows)	$325
	GraphiC 5.0 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	$325
	Drasch Clisp with Crules (Lisp library and programming environment with rule processing capability; natural language example)	$300
	PC Curses (Aspen, Software, System V compatible, extensive documentation)	$290
	C-Data Manager (object-oriented data management, persistent objects from runtime definitions, network and entity models)	$250
	MEWEL (extensible window and event library by Magma Software; message-passing & object-oriented; SAA-compatible; dialog editor)	$250
	TurboTEX (Release 2.0; HP, PS, dot drivers; CM fonts; LaTEX; MetaFont)	$250
	PC PostScript (complete PostScript interpreter (ROM Version 47.0A), 80286/386 only, many device drivers, optimized graphics, fast)	$250
	db_File & db_Retrieve by Raima (B-tree and network database with SQL query and report writer; multi-user $475)	$245
	Greenleaf Communications Library (interrupt mode, modem control, XON-XOFF; specify compiler)	$225
	CDirect (multi-user hashed file manager; variable length fields, binary or ASCII data, alternate keys)	$210
NEW!	SilverComm (complete asynchronous communications library)	$210
	Installation Toolkit 3.0 (interactive script development program; many special cases covered)	$195
	QuickGeometry Library (large collection of mathematics, graphics, display & DXF subroutines for CAD/CAM/CAE/CNC)	$170
	CBTree (B+tree ISAM driver, multiple variable-length keys)	$165
	TurboGeometry (library of routines for computational geometry, Version 3.0)	$160
	AT BIOS Kit (roll your own BIOS with this complete set of basic input/output functions for ATs)	$160
	WKS Library Version 2.01 (C program interface to Lotus 1-2-3, dBase, Supercalc 4, Quatro, & Clipper)	$155
	OS/88 (industrial-strength U**x-like operating system, many tools, cross-development from MS-DOS)	$150
	Cephes Mathematical Library (over 100 high-quality, double-precision scientific functions)	$150
	ME Version 2.1 (programmer's editor with C-like macro language by Magma Software; Version 1.31 still $75)	$140
	Vmem/C (virtual memory manager; least-recently used pager; dynamic expansion of swap file)	$140
	Turbo G Graphics Library (all popular adapters, hidden line removal)	$135
NEW!	Rogue Wave Vector & Matrix Classes (inc. C++ overloadings for standard operators, matrix inversion & FFT; Zortech or GNU C++)	$125
	Power Search by Blaise Computing (regular-expression compiler; generates machine code on the fly)	$120
	Install 2.3 (automatic installation program; user-selected partial installation; CRC checking)	$120
NEW!	B-Strings (dynamic string handling; cut, copy, paste, search, user input, etc.; non-fragmenting memory management)	$105
	TE Editor Developer's Kit (full screen editor, undo command, multiple windows)	$105
	Minix Operating System (Version 1.3; U**x-like operating system, includes manual)	$105
Updated!	PC/IP (CMU/MIT TCP/IP for PCs; Ethernet, Appletalk & NETBIOS drivers, RVD, gateways)	$100
	B-Tree Library & ISAM Driver (file system utilities by Softfocus)	$100
	The Profiler (program execution profile tool)	$100
	QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library)	$90
Cheaper!	Booter Toolkit (floppy disk bootstrap routines, DOS file system, light-weight multitasking, windows, fast memory management)	$85
	Otter 1.0 (beautiful theorem-prover by Bill McCune; includes manual & two books by Wos; complete starter kit)	$80
	JATE Async Terminal Emulator (includes file transfer and menu subsystem)	$80
	MultiDOS Plus (DOS-based multitasking, intertask messaging, semaphores)	$80
NEW!	HY-PHEN-EX (a hyphenator for American English with over 4,800 rules)	$75
	Make (macros, all languages, built-in rules)	$75
	eval() (C function to evaluate ASCII infix expression string; 17 built-in functions)	$75
	XT BIOS Kit (roll your own BIOS with this complete set of basic input/output functions for XTs)	$75
	Professional C Windows (lean & mean window and keyboard handler)	$70
	Heap Expander (virtual memory manager using expanded memory, extended memory, and disk space)	$65
	Quincy (interactive C interpreter)	$60
	Symtab/Ptree (general-purpose symbol table/parse tree construction and management package; specify Symtab or Ptree)	$60
	Coder's Prolog (Version 3.0; inference engine for use with C programs)	$60
	Async-Termio (Unix V compatible serial interface for MS-DOS; stty, ioctl, SIGINT, etc.)	$55
	Backup & Restore Utility by Blake McBride (multiple volumes, file compression & encryption)	$50
NEW!	Floppy TAR (TAR backup and restore on MS-DOS devices; direct access to non-standard devices)	$50
	SuperGrep (exceptionally fast, revolutionary text searching algorithm; also searches sub-directories)	$50
	REGX Plus (search and replace string manipulation routines based on regular expressions)	$50
	OBJASM (convert .obj files to .asm files; output is MASM compatible)	$50
	Multi-User BBS (chat, mail, menus, sysop displays; does not include Hayes modem driver)	$50
	LaplaceB (LaPlace polynomials, real and complex)	$50
	CLIPS (rule-based expert system generator, Version 4.3; advanced manuals available)	$50
	Kier DateLib (all kinds of date manipulation; translation, validation, formatting, & arithmetic)	$45
	Fortran-to-C Translator by Polyglot (Fortran-IV-like Fortran to ugly C; plan to adapt to your own flavor of Fortran)	$40
	DES Encryption & Decryption (2500 bits/second on 4.77 MHz PC for on-the-fly encryption at 2400 baud)	$40
	FlexList (doubly-linked lists of arbitrary data with multiple access methods)	$40
	Virtual Memory Manager by Blake McBride (LRU pager, dynamic swap file, image save/restore)	$40
	Heap I/O (treat all or part of a disk file as heap storage)	$40
Updated!	Bison & BYACC (YACC workalike parser generators; documentation; no restrictions on use of BYACC output)	$35
	PC-XINU (Comer's XINU operating system for PC)	$35
	RXC & EGREP (Regular Expression Compiler and Pattern Matching; RXC makes finite state machine from regular expression)	$35
	CCALC (handy extended-precision calculator; real and complex models; many built-in functions)	$30
	GNU Awk & Diff for PC (both programs in one package)	$30
Updated!	6-Pack of Editors (baker's half-dozen public domain editors for use, study & hacking; includes microEmacs 3.10 & Stevie, a vi clone)	$30
	Crunch Pack (14 file compression & expansion programs)	$30
	Pascal P-Code Compiler & Interpreter or Pascal-to-C Translator (Wirth standard Pascal)	$25
	FLEX (fast lexical analyzer generator; new, improved LEX; official BSD Version 2.1 with docs)	$25
	List-Pac (C functions for lists, stacks, and queues)	$25
NEW!	Using C++ Library (the code from the book by Bruce Eckel and then some; Zortech 2.0 compatible)	$25
	A68 (68000 cross-assembler)	$20
	XLT Macro Processor (general purpose text translator)	$20
	Data	
	Moby Pronunciator (150,000 words & phrases encoded with full IPA pronunciation & emphasis points; 900 distinguished by part-of-speech)	$160
	Moby Part-of-Speech (200,000 words and phrases described by prioritized part(s)-of-speech)	$120
	Moby Hyphenator (150,000 words fully hyphenated/syllabified)	$105
	Moby Words (500,000 words & phrases, 9,000 stars, 15,000 names)	$65
	Smithsonian Astronomical Observatory Subset (right ascension, declination, & magnitude of 258,997 stars)	$60
	U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	$35
	The World Digitized (100,000 longitude/latitude of world country boundaries)	$30
	KST Fonts (13,200 characters in 139 mixed fonts; specify TEX or bitmap format)	$30
Updated!	Interactive Computer Ephemerii (high-precision moon, sun, planet & star positions; USNO (no source) & Downey 4.8 (C source))	$30
	U. S. Map (15,701 points of state boundaries)	$15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3409 USA

info@acw.com

Voice: (512) 258-0785
BBS: (512) 258-8891
FAX: (512) 258-1342

Free surface shipping for cash in advance **For delivery in Texas add 7%** **MasterCard/VISA**

Number 30 on the Reader Service Card

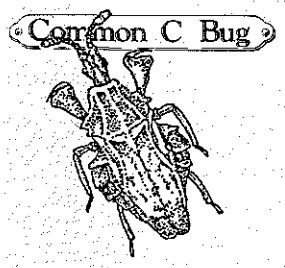
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

T-SHIRTS

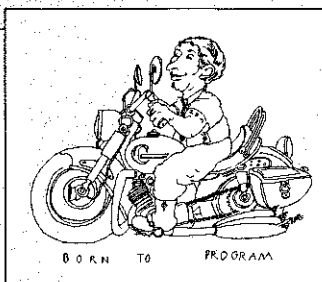
Multi-color imprints on the highest quality 100% cotton T-shirts.

SIZES: S, M, L, XL, XXL

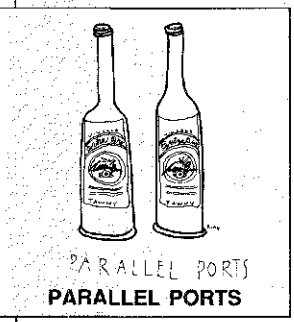
$12.00 each – includes shipping & handling



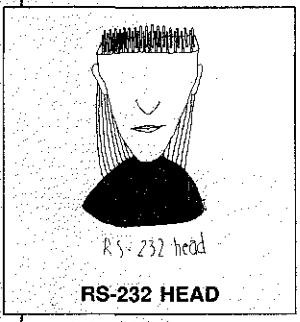
COMMON C BUG



BORN TO PROGRAM



PARALLEL PORTS



RS-232 HEAD

BUGS RULE 4 color imprint on a white shirt

PJ ISSUE SPECIAL!

SAVE $2.00 ON EACH BACK ISSUE

ALL BACK ISSUES NOW ONLY $5.00 EACH!

For a limited time, you can save over 28% on each back issue you order for your library. Complete your collection today—order while supplies last!

(Offer expires April 30, 1990. All orders must be prepaid.)

BACK ISSUES

~~$7.00~~ $5.00 each PREPAID ONLY

A few of the Back Issues currently available

Vol. 6.3: Hal Hardenbergh on memory trends; Pink Noise; getting your consulting tools in order; Gordon Letwin on OS/2 memory management; virtualizing the 386; Pascal—a CASE of the jumps; Turbo Professional 4.0; BASIC—CALL, GOSUB, and multiline functions.

Vol. 6.4: Job shopping; windowing systems overview; reading VGA memory; X Window—a view from DOS; a digital clock in C for MS Windows; windowing in TopView, DESQview, and MS Windows; MS C 5.1 and the ANSI standard; a mouse unit for Turbo Pascal 4.0.

Vol. 6.5: Hal Hardenbergh on computers in the fast lane; Consulting—no-win jobs and too much work; VGA text paging; using dynamic arrays and records in Turbo Pascal 4.0; an OS/2 Presentation Manager application template; WATCOM C 6.0 and the ANSI standard; object-oriented OS/2 in C++; Ada and software engineering; REXX on the PC.

Vol. 6.6: PC DOS 4.0; OS/2 vs. UNIX; VGA screen saving and color selection; MASM quirks; addressing extra memory for DOS; DOS device drivers debugging and information utility; proposed ANSI C language standard; Ray Duncan on Forth; Ada data types and exceptions; PC floating-point hardware and software.

Vol. 7.1: GRAPHICS: High-resolution VGA color modes; graphics processors; modifying type constants stored in Turbo Pascal 4.0 and 5.0 compiled EXE code; comparing Pascal overlay managers; JPI's TopSpeed Modula-2; faster QuickBASIC code; Microsoft's CodeView debugger; Lattice C V3.3; smart terminal emulations in Windows; the COBOL-85 ANSI standard; Pink Noise—industry news and reviews.

Vol. 7.2: OPERATING SYSTEMS: undocumented DOS features; fast 80386 block moves under DOS; OS/2 dynamic link libraries; OS/2 multitasking; debugging utility for PM and Windows; EGA and VGA animation; disk formatting in Pascal; how to become a reseller; reviews of windowing and graphics tools, disassemblers, and *The 80386 Book*.

Vol. 7.4: COMMUNICATIONS: "Pink Noise," by Frank Greco; Michael Abrash's "On Graphics"; a simple terminal program in PM; communications and protocols; better coding conventions for assembly language; implementing the Boyer–Moore search algorithm in assembly language; database normalization theory and practice; Turbo Pascal 5.5 development tools; *Lint utilities for the PC*; a *millisecond timer for BASIC*; *exit codes for* QuickBASIC; reviews of RamNet, DOS/16M, PCX *Technical Reference Manual* and Programmer's Toolkit, GRASP 3.1, AXOS, and *Microsoft Mouse Programmer's Reference*.

Vol. 7.5: OBJECT-ORIENTED PROGRAMMING: Class libraries in C++; from C to objects; improved C++ code generation; Hal Hardenbergh on the RISC/CISC schism; handling version management on large projects; "Still-Higher 256-Color VGA Resolutions," by Michael Abrash; a library system for assembly language; adding world scaling to Turbo Pascal's BGI graphics primitives; numeric extensions to C; reviews of 386 development environments and tools, mini-case tools, and Robert Sedgewick's book *Algorithms*.

Vol. 7.6: USER INTERFACES: Windows terminal program; the CURSES C library; using two video displays with Turbo Pascal; Michael Abrash on VGA and EGA split screens; memory models and static variables for assembly language; ANSI C standard; Aztec C version 4.2; using object-oriented programming to create a file viewer; accessing large memory spaces from DOS on the 80386; making use of QuickBASIC's string-searching algorithm; reviews of user-interface tools and mini-Case tools; the *System BIOS for IBM PC/XT/AT*, series, by Phoenix Technologies.

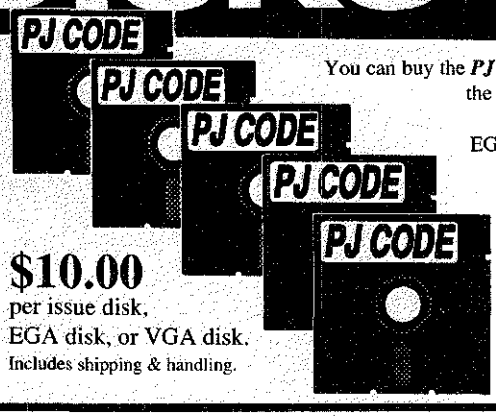
Vol. 8.1: NETWORKING: Frank Greco on the Network File System; an overview of NetBIOS on the PC, excerpted from *C Programmer's Guide to NetBIOS*, by W. David Schwaderer; fast circle drawing on the VGA, by Michael Abrash; "Dynamic Data Exchange in Windows," by William S. Hall; an assembly language program for effective use of dynamic memory, by John Otken; taking advantage of DESQview API features; computing time in C; Tom Swan on creating a menu system with Turbo Pascal 5.5; government agencies as consulting customers.

10% discount on any order of 10 or more T-Shirts, Disks or Back Issues.

All mail orders must be accompanied by check or Money Order -- PREPAID ONLY.

CODE DISKS

Use the order form on the card opposite.
Order by phone with your VISA/MC.
CALL TOLL FREE
800/234-0386 ext 53



You can buy the *PJ* code listings from volume 5.5 to the current issue. Code disks for the Michael Abrash "On Graphics" EGA/VGA series are also available.

$10.00 per issue disk, EGA disk, or VGA disk. Includes shipping & handling.

Diskettes are available for the following issues:

5.5	6.3	7.1	7.5	EGA
5.6	6.4	7.2	7.6	VGA
6.1	6.5	7.3	8.1	
6.2	6.6	7.4	8.2	

360K MS-DOS disks standard; other formats available.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

THE INTERNATIONAL NEWSLETTER FOR C++ PROGRAMMERS

THE C++ REPORT

Plus a whole lot more.

Stimulates, tracks & evaluates the growth of the C++ language

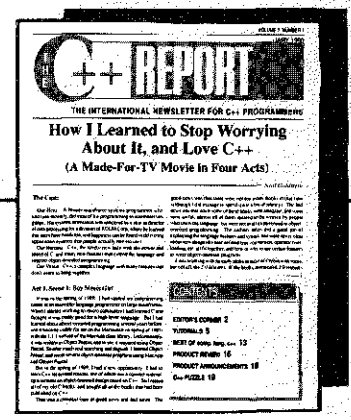
Get plugged into the "insiders" network. Keep on the cutting-edge of C++ development. The only independent publication in the world devoted exclusively to C++. Now in its second year.

Written for both experienced users and those wanting to learn C++. Written by our international staff of C++ programming experts — many of whom are pioneers in the language.

Fast-reading, ground-breaking, code-intensive, up-to-the-minute.

THE C++ REPORT

Links you with the best minds in C++.
Ten times a year you get:


C++ REPORT
THE INTERNATIONAL NEWSLETTER FOR C++ PROGRAMMERS
How I Learned to Stop Worrying About It, and Love C++
(A Made-For-TV Movie in Four Acts)
CONTENTS

- **FEATURE ARTICLES** — written for knowledgable users
  - "Standardizing C++"
  - "Type parameterization"
  - "C++ and Physics Computation"
  - "Managing C++ Libraries: Subdirectories and C Files"
  - "The Importance of Being Abstract"
  - "C++ As Close as Possible to C— But No Closer"
  - "Well-Behaved type relationships"
  - "Using C and C++ together"
  - "New Qualifiers for Member Functions in Release 2.0"
- **TUTORIALS** — written for new users
  - "References"
  - "Using In-line Functions Effectively"
  - "Implementing a Lexer in C++"
  - "Avoiding copies of temporary variables"
- **PRODUCT NEWS** — get the scoop on the latest C++ products and services as well as general industry news.
- **PRODUCT & BOOK REVIEWS** — candid & hard-hitting — saves you time and monev
- **BEST OF COMP. LANG. C++** — a digest of comp. lang. C++ traffic
- **EDITORIAL** — Editor Robert B. Murray, AT&T Bell Labs discusses controversial issues.
- **C++ PUZZLE** — so you think you're a good programmer? We challenge you to solve a given problem in as few lines as possible (or some other best way)
- **ADVERTISING** — limited in size and restricted to C++ related products
- **HELP-WANTED ADS** — career opportunities for C++ users.

Faster Service call:
Phone: 212-972-7055
Fax: 212-370-4983

Order Form

☐ ***YES,*** I want to stay abreast of the latest ideas in C++. Enter my subscription and rush me the current issue. This is a *risk-free* offer. I may cancel at any time and receive a prompt refund of the unused portion.

	1 year - 10 issues	*2 years - 20 issues*
Domestic	☐ $69	☐ $128
Foreign & Canada	☐ $89	☐ $168

☐ **Payment enclosed** - payable to ***The C++ Report.***
Foreign orders must be prepaid in U.S. dollars drawn on a U.S. bank

☐ **Bill Me**

☐ **Charge my:** ☐ Visa ☐ Master card

Card# _____ Exp. date _____
Name _____
Company _____
Address _____
City _____ State _____ Zip _____

Return to:
The C++ Report, 310 Madison Avenue, Suite 503, New York, NY 10017

Number 205 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Continued from page 86

latest version of Dis.Doc Professional costs $149.95.

RJSwantek, Inc.
178 Brookside Rd.
Newington, CT 06111
203-953-0236

Scientific Packages Updated by SciTech

SciTech Software has released version 3.10 of both their Scientific Package (SSP) and Plotting Package (SPP). These FORTRAN routines provide an integrated scientific library and plotting package for use with Microsoft FORTRAN (8086/80286) and SVS FORTRAN on both the 386 and 680X0 hardware (including Definicon and Yarc PC coprocessor boards). The scientific library has added new sorting routines, serial port initialization, and Super-VGA support for plotting routines. The plotting package that emulates Calcomp syntax now supports a number of Super-VGA boards along with new printers and multipen plotters. Each package costs $225 until March 31, when the price increases by $100. Upgrades from earlier versions are $50.

SciTech Software
5354 Palm Drive
La Canada, CA 91011
818-790-3844

Fresh Crop of Compilers in from the Valley

Silicon Valley Software has released revised versions of their C, Pascal, and FORTRAN compilers for the 386. These revised versions now work under Phar Lap's DOSIExtender and Virtual Memory Manager. The SVS symbolic debugger is also available for program development with any of these languages. Similar compilers are also available for SUN 386i and SUN/3 machines running SunOS. The new DOS compilers are $825 for FORTRAN and $725 each for Pascal and C. The debugger is now sold separately for $325, and Weitek support costs an additional $170.

Silicon Valley Software
1710 South Amphlett Blvd., Suite 100
San Mateo, CA 94402
415-572-8800

Sourcer Updates Commenting Disassembler

V Communications has released version 2.2 of Sourcer 486, a commenting disassembler. Sourcer creates detailed, commented source code and listings from memory and executable files. Using a built-in data analyzer and code simulator, Sourcer identifies segments and resolves data across multiple segments. Detailed comments are provided on interrupts, I/O ports, and DOS and BIOS functions. A definition-file facility allows the user to include remarks and descriptive labels, force data types, and so on. Sourcer has now been upgraded to include complete support for the 8088 through 80486 and the 8087 through 80387 instruction sets. Other major additions include automatic definition-file generation and up to nine passes of analysis. V Communications has also upgraded BIOS Pre-Processor, which can now generate detailed source listings for the latest 80386 and 80486 BIOS ROMs. An Unpacker utility is also available to unpack EXE files for disassembly. Sourcer costs $99.95 (139.95 with BIOS Pre-Processor), and the Unpacker is $39.95.

V Communications, Inc.
4320 Stevens Creek Blvd., Suite 275
San Jose, CA 95129
800-662-8266

The Software Engineering Store

C

Borland Turbo C v2 Prof. $175
Blaise C Tools Plus/6.0 ..... $119
Clear Clear Plus for C $169
Creative Vitamin C $169
Genus PCX Prgmr Toolkit $155
manipulate ".pcx" graphics files
Gimpel PC Lint $99
Greenleaf Comm Library $179
Island Systems C Menu $129
Microsoft C v5.1 ..... $325
Microsoft Quick C $73
Novell Btrieve $199
Sequitor Codebase 4 ..... $249
read, write & create dBase III and Clipper files
Wordcraft C Workshop $69
C training manual & software

Other Tools

Digitalk Smalltalk/286 ..... $164
Burton Tlib $89
source code management system
California Scientific Brainmaker $159
neural network development system
Genus PCX Toolkit $155
Helios Proteus $169
Matrix Layout v2 ..... $175
Microsoft Macro Assembler $111
Microsoft Windows SDK v2.1 ..... $349
Mortice Kern MKS Toolkit v3 ..... $219
KORN shell & 130 UNIX™ -like utilities
Periscope I 512K v4 ..... $499
Qualitas 386-Max $69
80386 memory manager
Quarterdeck API Ref & C Library .. $179
Solution Systems Brief ..... $call
Turbo Power Turbo Professional $99
Whitewater ACTOR $449
High productivity development for MS Windows

C++ Version 2

Zortech C++ is the only true C++ compiler available for MSDOS. Now includes AT&T C++ v2 features including multiple inheritance and type safe linking. C++ debugger lets you view objects instead of translated C code.
C++ v2 ..... $175
C++ Debugger $129
C++ Developer's Edition $399

The Software Engineering Store

1-800-443-7176

In Calif: (415)957-0111

Visa • MasterCard • AMEX
COD • Check • P.O.

- Most orders shipped within 24hrs
- Free shipping on orders over $100

657 Mission Street, #300-PJ
San Francisco, CA 94105

Free Newsletter

Call or write for a catalog and your free subscription to our newsletter.

Prices subject to change without notice ©1989 TSF All Rights Reserved

Number 98 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

SOFTWARE DEVELOPERS: ARE YOUR PRODUCTS TOO HARD TO INSTALL?

In today's competitive market, your product **must** be easy to use -- immediately. It is no longer commercially acceptable to tell the end-user to use batch files, COPY, DISKCOPY, RESTORE, or to require wading through paragraphs of text in the manual. The installation process is something that a program developer **should** and now **can easily** completely automate.

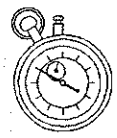
INSTALL 2.3 is a bullet-proof automatic installation program you can distribute - royalty free - with your product • Features: Up to 4.3 gigabyte file sizes • Windowing/color user interface • **Full C source** with project, make, link, and batch files • File integrity verification • Intelligent handling of virtually all errors (open disk drive doors, full disks, unformatted disks, etc) • Hard & floppy disk installation • Multiple source & target disks • Multiple target subdirectories (which *INSTALL* will create if necessary) • Unlimited free tech support • Granular installation • Easy method to add-in new functionality • Ability to execute DOS commands • Can intelligently modify CONFIG.SYS & AUTOEXEC.BAT files • Includes **data compressing file librarian** • **NEW** Automatically detects and can base the installation on the CPU (86/286/386), DOS version, display (CGA/B&W/MONO/EGA/VGA), NetBIOS, ANSI.SYS, LIM, etc.

INSTALL offers 3 years of proven reliability that has been used to install over 500,000 copies of products including some of the most prestigious in the industry. We would like to add your name to that list.

ORDER NOW TOLL FREE 1-800-331-2783

KNOWLEDGE DYNAMICS CORPORATION
Highway Contract 4 Box 185-H, Canyon Lake, TX 78133-3508
Voice: 1-512-964-3994 FAX: 1-512-964-3958
MasterCard/VISA/COD/Purchase Order **$149.95**

Number **295** on the Reader Service Card



15 Minutes to Success!

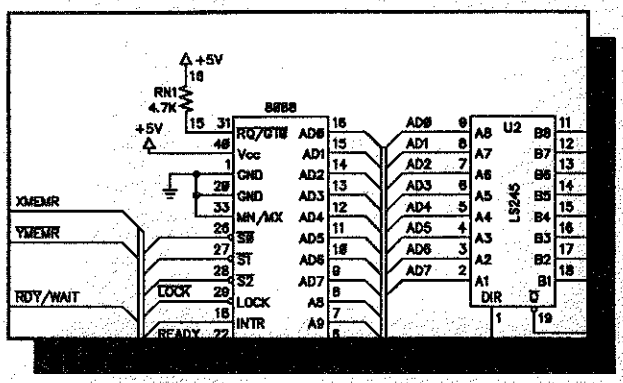
Install and learn our project management software in fifteen minutes . . . or your money back!

SPMT is a complete project management environment tailored to software development projects. Ease of use without sacrificing power has been our objective. Handles multiple projects in one or many files. Enter up to 10,000 tasks in your WBS, then collapse the WBS to get an overview. Use a mouse to click on WBS elements to make changes. From the Gantt chart, use the mouse to drag tasks to their new start date. Full support for the government's DOD-STD-2167A and C/SCSC cost accounting. Includes LAN support. Site licenses and source code are available. The complete package retails for $492. Orders are shipped UPS COD. To order, call or write today!

W.H. Roetzheim & Associates
13518 Jamul Drive, Jamul, CA 92035
(619) 669-6970

Number **339** on the Reader Service Card

WALL-SIZE XT SCHEMATIC



At last you can plumb the mysteries of your computer with this single sheet (24" x 36") schematic of the IBM XT's main board. This is a wealth of information for both True Blue *and* clone owners.

This is the kind of information you need for tracking down board problems, for designing add-on cards, or for turning XT motherboards into inexpensive controllers/monitors. And, because this is a single-sheet schematic, you won't spend hours tracking signals from page to page. You see the entire motherboard at once.

Trouble-Shooting Information

As a bonus, we include trouble-shooting information. For instance, your board is dead but a quick look at the schematic shows you the location of PPI port A. You know that the power-on self test loads a number into port A each time it completes a self test. All you need to do is read the port with a logic probe to see how far the system went before it discovered an error. We include these checkpoints and other trouble shooting information with the schematic.

NOTE: Although clones often differ a bit from IBM in layout and IC designations, their circuits are close enough to the original XT for this schematic to be very useful.

IBM PC-XT Schematic $15.00
U.S. shipping and handling $1.95
Foreign (airmail shipping and handling) $3.95

THE MICRO TECHNICAL JOURNAL

MICRO CORNUCOPIA

Dept PJ, P.O. Box 223, Bend, Oregon 97709

Monday-Friday, 9AM-5PM PST

1-800-888-8087

FAX: 1-503-389-6833 (24 hours)

VISA MC

Number **350** on the Reader Service Card

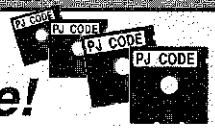
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

TOPIC	VOLUME	AUTHOR	TITLE
386 Development	7.2	Baker, M. Steven	386 Now
386 Development	7.5	Baker, M. Steven	Soft-ICE and MagicCVW (Nu-Mega Technologies)
386 Development	7.5	Baker, M. Steven	FTN77/386 (OTG Systems)
386 Development	7.6	Roden, Thomas	Four Gigabytes in Real Mode
ASM	7.4	Otken, John	A Line of Code
ASM	7.4	Crum, Adrian B.	Fast String Searching
ASM	7.5	Otken, John	A Library System
ASM	7.6	Otken, John	Efficient 80X86 Memory Models and Static Variables
BASIC	7.1	Winer, Ethan	Which Code is Faster?
BASIC	7.1	Baker, M. Steven	QuickPak Professional (BASIC utilities)
BASIC	7.4	Winer, Ethan	A Millisecond Timer for BASIC
BASIC	7.4	Lesser, Murray L.	Exit Codes for QuickBASIC
BASIC	7.6	Winer, Ethan	QuickBASIC's Fast String-Searching Algorithm
Books	7.2	Baker, M. Steven	The 80386 Book, by Ross Nelson (Microsoft Press)
Books	7.3	Baker, M. Steven	A Guide to the SQL Standard, by C. J. Date (Addison-Wesley)
Books	7.4	Baker, M. Steven	Microsoft Mouse Programmer's Reference (Microsoft Press)
Books	7.5	Baker, M. Steven	Algorithms, by Robert Sedgewick (Addison-Wesley)
Books	7.6	Baker, M. Steven	System BIOS for IBM PC/XT/AT series, by Phoenix (Addison-Wesley)
C	7.1	Jaeschke, Rex	The Proposed ANSI C Language Standard
C	7.2	Jaeschke, Rex	The Proposed ANSI C Language Standard
C	7.3	Jaeschke, Rex	The Proposed ANSI C Language Standard
C	7.5	Jaeschke, Rex	Numeric Extensions, Zortech C++, and ANSI C
C	7.6	Jaeschke, Rex	Aztec C and the ANSI C Standard Delayed
COBOL	7.1	Riehle, Richard	COBOL Comes of Age
Case Tools (Mini)	7.5	Baker, M. Steven	CC-RIDER (Western Wares)
Case Tools (Mini)	7.5	Baker, M. Steven	EasyFlow (HavenTree Software)
Case Tools (Mini)	7.6	Baker, M. Steven	Logic Gem (Sterling Castle)
Communications	7.4	Baker, M. Steven	PC Communications and Protocols
Communications	7.4	Baker, M. Steven	RamNet (Software Concepts Design)
Conferences	7.1	Baker, M. Steven	Usenix and C++
Conferences	7.3	Baker, M. Steven	Software Development '89
Consulting	7.2	Richter, Jake	Reselling
Consulting	7.5	Cunliffe, Ken	Source Code Version Control
Consulting	7.6	Roetzheim, William H.	Wealth Through Project Growth
DOS	7.2	Michels, Raymond J.	Undocumented DOS Internals
DOS Extenders	7.5	Baker, M. Steven	OS/286 and OS/386 Developers' Kits (Eclipse)
DOS Extenders	7.4	Baker, M. Steven	DOS/16M (Rational Systems)
Database	7.3	Riehle, Richard	Embedded SQL
Database	7.3	Baker, M. Steven	Converting Database Files Using Filters
Database	7.3	Baker, M. Steven	CodeBase (Sequiter Software)
Database	7.3	Baker, M. Steven	Apex Database Library (Apex Software)
Database	7.4	Madison, Bill	Database Normalization
Disassemblers	7.2	Baker, M. Steven	SOURCER (V Communications)
Disassemblers	7.2	Baker, M. Steven	snOOp II (TriDOS Software)
Disassemblers	7.2	Baker, M. Steven	MD86 (C. C. Software)
Disk Formats	7.2	Newlin, John	Disk Formatting
FORTRAN Tools	7.3	Baker, M. Steven	Prism Toolkit (OTG Systems)
File Managers	7.3	Khalsa, Amrit and Maclean, Pete	Btrieve for C Programmers
File Managers	7.4	Baker, M. Steven	AXOS (McSoft Incorporated)
Graphics	7.1	Abrash, Michael	Higher 256-Color Resolution on the VGA
Graphics	7.1	Richter, Jake	Migrating to Graphics Processors
Graphics	7.2	Abrash, Michael	EGA and VGA Animation
Graphics	7.3	Abrash, Michael	Fast Line Drawing for the EGA and VGA
Graphics	7.4	Baker, M. Steven	PCX Technical Reference Manual (Zsoft)
Graphics	7.4	Baker, M. Steven	PCX Programmer's Toolkit (Genus Microprogramming)
Graphics	7.4	Baker, M. Steven	GRASP 3.1 (Paul Mace Software)
Graphics	7.5	Abrash, Michael	Still-Higher 256-Color VGA Resolutions
Graphics	7.6	Abrash, Michael	VGA and EGA Split Screens
Modula-2	7.1	Newlin, John J.	From Pascal to Modula-2
Networking	7.1	Baker, M. Steven	GOSIP—The Government's OSI Network Standard
Neural Nets	7.3	Waite, Tom and Hardenbergh, Hal	Neural Nets

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

TOPIC	VOLUME	AUTHOR	TITLE
OS/2	7.2	Duncan, Ray	OS/2 Dynamic Link Libraries
OS/2	7.2	Vernon, Vaughn	OS/2 Multitasking with Class
OS/2	7.4	Hall, William S.	A Simple Terminal Program in Presentation Manager
Object-Oriented (C)	7.5	Sando, Steve	From C to Objects
Object-Oriented (C++)	7.3	Vernon, Vaughn	User Interface Tools for DOS
Object-Oriented (C++)	7.5	Vernon, Vaughn	The Forest for the Trees
Object-Oriented (C++)	7.5	Dlugosz, John M.	Better C++ Code Generation
Object-Oriented (Pascal)	7.6	Swan, Tom	Using OOP to Create a Powerful File Viewer
Pascal	7.1	Swan, Tom	Demagnetizing Static Variables
Pascal	7.1	Madison, Bill	My Kingdom for an Overlay
Pascal	7.4	Swan, Tom	The Future Arrives
Pascal	7.5	Fowler, Gene	The World in Your Hands
RISC	7.5	Hardenbergh, Hal W.	A Technological Schism
Software Tools	7.1	Baker, M. Steven	OPUS Make 5.1 (DOS and OS/2)
Software Tools	7.3	Baker, M. Steven	Optlib and Optlib/2 (SLR Systems)
Software Tools	7.3	Baker, M. Steven	Inside! (Paradigm Systems)
Software Tools	7.3	Baker, M. Steven	ASMFLOW (Quantum Software)
Software Tools	7.4	Abrash, Michael	Measuring Performance
Software Tools	7.4	Jaeschke, Rex	Lint Utilities
UNIX	7.1	Greco, Frank D.	Pink Noise
UNIX	7.4	Greco, Frank	Pink Noise
User Interface	7.6	Vernon, Vaughn	Human Interfaces with Curses
User Interface	7.6	Cole, Guy	Cooking with Stereo Video
User Interface	7.6	Baker, M. Steven	Proteus (Helios Software)
Windowing/Graphics	7.2	Baker, M. Steven	XVT 1.2 (Advanced Programming Institute)
Windows	7.1	Hall, William S.	Smart Terminal Emulation in Windows
Windows	7.3	Hall, William S.	A Substitute for the Missing TYPE Command in Windows
Windows	7.6	Hall, William S.	A Simple Terminal Program for Windows
Windows/Graphics	7.2	Baker, M. Steven	TIGA-340 and TI Development Tools
Windows/PM	7.2	Hall, William S.	A Simple Debugging Utility for Presentation Manager and Windows

PJ Code Is Now Available!



On disk from:

Programmer's Journal
P. O. Box 70167
Eugene, OR 97401-0110

Call toll free:
800-234-0386

In Oregon call:
503-747-0800

All Code Disks $10.00 each.
To order, see ad on page 88.

***PJ* Code DiskScriptions are available for only $39.95.**
To order, see ad on page 8.

On BBS from:

Buffer Board
538 Cardinal Way
Springfield, OR 97477
(Voice) 503-741-1414
(Data) 503-747-7636

EDI
1332 Hermosa Ave. #7
Hermosa Beach, CA 90254
213-374-3478
Call for your local access number.

Generation Five Inc.
12061 Tech Road
Silver Spring, MD 20904-9998
(Voice) 301-680-3600
(Data) 301-680-9300
Accessible via PC Pursuit, Washington, D.C.
From Files submenu, select "P" for Personal Downloads.

Gilmore Systems of Southern California
5447 Indian Hills Drive
Simi Valley, CA 93063
(Voice) 805-582-1360
(Data) 805-582-9306

Ithaca Street BBS
1145 Ithaca
Boulder, CO 80303
(Voice) 303-494-8865
(Data) 303-494-8868

Logical Alternatives Incorporated
Calder Square,
P.O. Box 10674
State College, PA 16805-0674
(Voice) 814-234-8088
(Data) 814-234-1881

Mnematics Videotex
P.O. Box 19
Sparkill, NY 10976-0019
(Voice) 800-322-3633
(Voice) 914-365-0184 in N.Y.
Mention PJ when you call.

PCS-49 on CompuServe
Section 10
Sponsored by:
MISOSYS, Inc.
P.O. Box 239
Sterling, VA 22170
(Voice) 703-450-4181

The Silicon Zone
820 Tivoli Circle #203
Deerfield Beach, FL 33411
(Voice) 305-360-9362
(Data) 305-360-9376

COMING ISSUE—

May/June 1990 Volume 8.3

Operating Environments:

Accessing High Memory Under DOS
by Larry Smith

Undocumented Re-Entry to the DOS Shell
by Daniel Greenberg

Mouse Messages in Presentation Manager
by Chip Little

Graphics in the Windows Environment
by William S. Hall

X Window Under DOS
by M. Steven Baker

On Graphics:
Fast VGA Ellipse Drawing
by Michael Abrash

386 NOW:
Using 386 Debug Registers Under DOS
by John Otken

C:
Benchmarking C—Fact or Fiction?
by Rick Naro

Pascal:
More OOP for Turbo Pascal
by Tom Swan

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Offer your product directly to Professional Programmers:

Placing an ad in **PJ BILLBOARD** is an easy, low-cost way to reach potential buyers. Regular listings include a headline, company information, and approximately 400 characters for your message. Display ads are 1 5/8" wide by 2 3/8" high. For more information or to reserve a space, call our ad department at **503-747-0800.**

FIGHT PIRACY & PROTECT YOUR PROGRAM $$$'s!

Since 1986, companies worldwide have been choosing Az-Tech security products. If you demand the strongest protection available, why not choose one of these "proven leaders":

- **EVERLOCK** Copy Protection
- **EVERTRAK** Software Security
- **EVERKEY** Hardware "Key" Software Security

For IBM and Compatibles.
30 day money back guarantee.
Free info and demo disk available.

Az-Tech Software, Inc.
305 East Franklin
Richmond, MO 64085
(800) 227-0644 **(816) 776-2700**
FAX (816) 776-8398

Number **51** on the Reader Service Card

HUGE Professional Shareware Libraries

Comprehensive, indexed collections of all known Public Domain/Shareware utilities for professional developers, updated quarterly for latest products. Includes extensive disk based or printed directories with multiple search options. Each library designed specifically with the PC professional in mind. Computer Language called our dBase product "possibly the most productive $100 you'll ever spend". VISA, MC, POs, COD.

Product	Disks/Files	Price
dBase/dBase Compilers	48+/1000+	$99.50
AutoCAD	20+/600+	$99.50
Utilities for DOS Gurus	20+/200+	$49.50
C	20+/400+	$49.50
Turbo Pascal	15+/200+	$49.50
Ventura DTP	12+/150+	$49.50
Lotus 123	14+/250+	$49.50
Games	30+/200+	$49.50

EMS, 4505 Buckhurst Ct.
Olney, MD 20832 USA
(301) 924-3594 Fax: (301) 963-2708
In France: 14 7763872, Australia 62 852097

Number **310** on the Reader Service Card

EZ-COPY PLUS

Diskette Duplicator
2X+ faster than DISKCOPY!
- Converts your own XT/AT, etc.
- Makes DOS copies true and fast!
- Serializes, saves images to Hard Drive
- Speed optimization & partial formatting.
- **All 5.25" & 3.5" formats.**
- *ONLY* **$129** +S&H

Orders(V/MC/AX) & Brochures Toll Free:
1•800•873•2799 xPJ390
INFO 1-713-280-9900
S&H: $6 USA-48; $10 N.Amer; $20 Fgn.
EZX Publ., 917 Oakgrove Dr., Ste. PJ390, Houston, TX 77058 •FAX: 713-480-0525

Number **352** on the Reader Service Card

FREE CATALOG!

LATEST AWARD BIOS
PC/XT, 286, 386
COMPILED BASIC TOOLS
Libraries, Utilities
for QuickBASIC 4.5
HARDWARE UPGRADES
for PCs & PC/XTs
(800) 423-3400 - (412) 782-0384
KOMPUTERWERK INC.
851 Parkview Blvd.
Pittsburgh, PA 15215

Number **314** on the Reader Service Card

RELATIONAL DATABASE LIBRARY

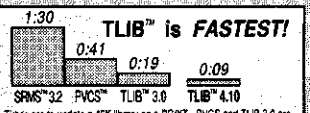
For TurboC and Microsoft C on DOS systems. The 114 functions in the **"c_wndw"** and **"c_ndx"** libraries support:

- **Multiple B-tree index files**
- **Stacked, overlapping windows**
- **dBase .DBF and .NDX file formats**
- **Multi-level file relationships**
- **Pull-down and pop-up menus**
- **Hashed random file access**
- **Data dictionary field access**

Libraries $99.50; Source $199.50; VISA/MC

Marietta Systems, Inc.
P.O. Box 71506
Marietta, GA 30007
(404) 565-1560

Number **223** on the Reader Service Card



Times are to update a 45K library on a PC/XT. PVCS and TLIB 3.0 are from Sept 87 *PC Tech Journal*. SRMS and TLIB 4.10 are later versions.

TLIB™ is *BEST!*

"Do not be fooled by the fact that this is the least expensive of the five packages reviewed here - TLIB has features and power to spare" John Rex, **Computer Language**

"TLIB is a great system" J. Valiino, **PC Tech J**

- **Full-Featured Version Control for Software Professionals.** Check-in/out locking. Branching. Keywords. Wildcard and list-of-file support. Can merge parallel changes and undo intermediate revisions. Network and WORM support. **Mainframe compatible** deltas for Pansophic, ADR, IBM, Sperry. Integrated Landon Dyer PD **MAKE.**

MS-DOS 2.x, 3.x, 4.x ***Just $99.95*** + $5 s/h Visa/MC
5 station LAN license $299.95 + $5 s/h, call for other sizes

BURTON SYSTEMS SOFTWARE
PO Box 4156, Cary, NC 27519 **(919) 856-0475**

Number **288** on the Reader Service Card

LAI

Logical Alternatives Incorporated

SMP's amazing windows, now available in *Basic*!

FREE DEMO

(814) 234-8088
BBS (814) 234-1881

Contact this advertiser directly

Advanced
Programmer's Books & Tools for MS-DOS & OS/2

Professional books, function libraries, and utilities for creating high-performance MS-DOS programs, TSRs, OS/2 programs, and Presentation Manager applications.

C and Assembly Language!
Please call or write for free catalog:

Michael J. Young
Young Software Engineering
20 Sunnyvale Avenue, Suite A,
Dept PJ02
Mill Valley, CA 94941
415/383-5354

Number **168** on the Reader Service Card

PASCAL OBJECT LIBRARIES
For Turbo 5.5 & Quick Pascal 1.0

Andrews Data Systems introduces OOP tools for Pascal. Prototype sophisticated user interfaces quickly and easily. Create complex simulations and real-time systems in a fraction of the time required for procedural languages.

Windows - Form/field support and real-time extensions (lifeline and animated bar graphs). Text and graphic mode support. Includes form builder and attribute select.

Remote Object Support - distributed processing over serial line or network. Includes serial library.

Concurrent Object Support - multi-task within your application.

$59 each or $89 with source. Serial library alone: $39 or $59 w/source. Windows/Remote/Concurrent: $149 or $199 with source. Add $5 S/H all orders. Check, MO, VISA/MC. No CODs.

ANDREWS DATA SYSTEMS
P.O. BOX 37123
DENVER, CO 80237

Number **318** on the Reader Service Card

epsilon programmer's editor has what you've been missing . . .

State of the art editor for DOS, OS/2, Xenix & Unix for PC/XT/AT's

Undo and Redo functions. Ultimate customizability with C-like extension language with source code to all commands. Compiler journaling. EMS support. Multi-tasking compiles. Unlimited file size, number of files. Full command line editing. Record keystroke sequences. Optional backups. Not copy protected.

Ask about our free trial!

$195 direct 60 day money-back guarantee

Lugaru Software Ltd.
5843 Forbes Avenue
Pittsburgh, PA 15217
(412) 421-5911
FAX: (412) 421-6371

Number **271** on the Reader Service Card

Network API Library

For *C* or *Pascal*, supports both *Microsoft* and *Turbo* compilers, Compatible with *IBM, 3Com, Banyan*, and *Novell* Networks. Includes DOS network functions, **NetBIOS, IPX/SPX,** Client/Servers, **Remote Procedure Calls,** Remote screens/windows. $99.95

NetWorkStation Utilities

Network Consulting, Distributed Processing, Classroom Screen Broadcasts, Work Groups, Device Sharing. NetBIOS/Netware $99.95

Information Software Associates
Box 2582, Iowa City, IA 52244
Call for more info **(319) 354-6804**

Number **294** on the Reader Service Card

COLLEGE DEGREES

. . . from fully-accredited Colleges and Universities. We help Computer Professionals avoid years of unnecessary class work. Get the job offers and recognition you deserve. Phone UDA for more information and our free booklet "Career Tactics and Strategies."

University Degree Advisory
☎ 800-765-7272

Number **311** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

TURBO PASCAL PROGRAMMERS

TP3, an easy-to-use preprocessor, permits usage of assembly language rather than in-line statements to generate in-line machine code. The assembly source is retained as commentary.

All 8086 instructions (aaa..jmp.. loop..xor) are supported using MASM syntax. Pascal source passes through unchanged.

A really useful addition to your utility library. 5.25" ds/dd (xt) floppy.

$39 (plus 7% sales tax in CA)

MAIKON SOFTWARE
P.O. Box 24467
San Jose, CA 95154
(408) 223-1166

Number **354** on the Reader Service Card

1000'S OF NEW COMPUTER BOOKS UP TO 40% OFF

New titles for the professionals including books for PCs and compatibles, MAC, UNIX, minis, and mainframes. Dos, OS/2, X Window, X Open, and UNIX V/386. Ada, C, C++, Modula-2, Pascal, LISP, FORTH, ASM, and other languages.

Desktop publishing, word processing, CAD, graphics, image processing, and hardware.

Send five 25¢ stamps for complete 70-page catalogue.

Computer Books
P.O. Box 1758
Solano Beach, CA 92075 **RS# 340**

dBASE FILE ACCESS FROM C

Code Base 4 is a library of C routines which gives complete dBASE and Clipper functionality and file compatibility. The routines correspond directly to dBASE commands. Create multi-user applications using pulldown menus, popup windows and data entry.

Code Base 4 supports DOS, SCO Xenix, OS/2 and Microsoft Windows.

$295 with Source! FREE DEMO.

Sequiter Software Inc.
P.O. Box 5659, Station L
Edmonton, Alberta
Canada T6C 4G1
403-439-8171
Fax: 403-433-7460 **RS# 269**

CSOURCE APPLICATION GENERATOR

C code WYSIWYG screen, report, & program generator. Library code may be enhanced for custom generation. Supports multi-user development & DOS 3.1/Novell networks. Produces small-sized DATA ENTRY, REPORT, MENU, INDEX REBUILD & FILE RESTRUCTURE programs with multiple screens & files. Very fast development & execution speeds. Requires 256K RAM, DOS 2.1 or later, Faircom Ctree or Novell Btrieve. For Turbo or Microsoft C 5.0. No royalties, ALL generated applications source is supplied.
$395.00 executable, $595.00 full source code. Multi-programmer licenses.

MBAC
13320 Castana Ave.
Downey, CA 90242
213-531-7259 **RS# 291**

Programmer's Journal

May/June 1990
Vol. 8.3

Advertising Space Reservation
Deadline:
March 2, 1990
Materials Due:
March 9, 1990

Please call *Programmer's Journal* to reserve your space.
503-747-0800

DO IT YOURSELF!

XenoCopy-PC

PC-DOS program lets your PC
Read/Write/Format
over 350 formats **$79.95**

XenoFont

high quality text screen printouts
$49.95

+$5.00 S/H + Tax if CA.

XenoSoft • (415) 644-9366
2210 Sixth Street,
Berkeley, CA 94710

Number **123** on the Reader Service Card

"QUICK & CLEAN" COBOL

The COBOL Report Activator (COBRA):

- Generates efficient, structured COBOL programs
- Is easy to learn and use
- Accepts your COBOL code and COPY members
- Produces audit trail reports
- Up to 5 output files per program
- Provides report options & site defined defaults
- Is available for Mainframe & PC

COBRA
Systems Software
David Olivadoti, P.O. Box 750817
Petaluma, CA 94975
(707) 763-5123

Number **281** on the Reader Service Card

QuickBASIC File Indexing

Creating B+ Tree indexes in QuickBASIC with **Index Manager™**. Access files randomly by full or partial key, or sorted forward or backward. One module performs all functions making **IM** very easy to use. Only indexes managed! You retain full control over your data. Assembler code and cache buffers make **IM** very fast. Demo on GEnie and CompuServ. Eight sample programs and 64 page Users Guide included.

Only $59 (+tax in CA).
VISA/MC accepted.

CDP Consultants
1700 Circo Del Cielo Drive
El Cajon, CA 92020
(619) 440-6482

Number **335** on the Reader Service Card

Double-decker
STAND
$124.99
Sturdy, tubular steel, enamel finished oak and putty.
Piggybacks two printers. Casters optional- $8.00.
24x22x41¾High

Single printer stands- tubular steel, oak top. * Includes casters. Discounts avail.
132 col. 24x14x26½H $73.00*
80 col. 16x14x26½H 59.99
CRT workstations/tables avail., call for prices. 214 /**299-6658**

TRI-STAR DISTRIBUTION
3816 Flamingo Ln., Irving, TX 75062

Number **155** on the Reader Service Card

Why you want BATCOM!

BATCOM is a batch file compiler that compiles your ".bat" files to ".exe" files to make them faster, more professional, and more capable. **BATCOM** extends DOS with new commands so you can read keyboard input, perform arithmetic, use subroutines, and much more. In addition, **BATCOM** protects your source code, and you can distribute your compiled programs without royalties. Only **$59.95**

ORDER TODAY!

Wenham Software Co.
5 Burley St.
Wenham, Ma. 01984
(508) 774-7036

Number **320** on the Reader Service Card

PC TIMER TOOLS

PCHRT is the definitive answer to execution profiling and embedded timer applications in a PC/MSDOS environment. 40 functions manage multiple microsecond timers, generate precision delays, and insert timers in virtually any interrupt. Profile BIOS, DOS, and EMS effortlessly! Supports TC, TP, MSC. Libs, manual, examples, and full source included.

$49.95 ppd USA, elsewhere add $4.00. VISA/MC.

RYLE DESIGN
P.O. Box 22
Mt. Pleasant, MI 48804
517-773-0587

Number **210** on the Reader Service Card

dCURSES

A full implementation of the UNIX Curses Library for MS-DOS.

MSC
Turbo C
Lattice C

Object $69.95
Source $149.95
VISA/Mastercard
Call for FREE Shareware Copy

HIGH ASPECT DEVELOPMENT
9821 South Pulaski
Evergreen Park, IL 60642
(312) 424-5997

Number **313** on the Reader Service Card

Midnight Engineering™

A new magazine for entrepreneurial hardware and software engineers that will encourage and challenge you to personally develop and market your own products.

- PRACTICAL ARTICLES
- INSIGHTFUL INTERVIEWS
- DETAILED PRODUCT REVIEWS

PJ readers are invited to get a FREE copy of ***Midnight Engineering*** by calling or writing to us.

Midnight Engineering
111 E. Drake Rd., Suite 7041
Fort Collins, CO 80525
303-491-9092

Number **272** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Dup-Tek

SOFTWARE DUPLICATION
High Quality — Low Prices
Fast Turnaround

- Guaranteed diskettes for sale
- Rigid quality controls
- 5.25" duplication as low as $.55 each
- Label production & application
- 100% satisfaction-guarantee policy

VISA — MASTERCARD — C.O.D.

DUPLICATION TECHNOLOGIES LTD.
P.O. Box 384
Odessa, MO 64076
816-633-8679 FAX: 816-633-8697

Number **338** on the Reader Service Card

The $25 Network

Try the 1st truly low cost LAN

- Connect 2 or 3 PCs, XTs, ATs, PS/2s
- Uses serial ports and 5 wire cable
- Runs at 115K baud, up to 90 feet
- *Transfer 8500 bytes per second (ATs)*
- Runs in background, totally transparent
- Share any device, any file, any time
- Needs only 14K of ram
- Replace all file transfer software
- $25 per network, NOT PER NODE!
- OVER 15,000 SOLD WORLD WIDE

Skeptical? We make believers!

iMODES Information Modes
Drawer F, Denton, TX 76202
817-387-3339 Technical
1-800-628-7992 Orders

Number **348** on the Reader Service Card

OPT-TECH SORT/MERGE

Extremely fast Sort / Merge / Select utility. Run as an MS-DOS command or CALL as a subroutine.
Supports most languages and filetypes including Btrieve and dBase. Unlimited filesizes, multiple keys and much more!

MS-DOS **$149.**
OS/2, UNIX **$249.**

Opt-Tech Data Processing
P. O. Box 678
Zephyr Cove, NV 89448
(702) 588-3737

Number **16** on the Reader Service Card

BTRIEVE ALTERNATIVE

Application programmers: Get a secure, multilanguage, multiuser, full, flexible indexed file manager: AXOS, only $395. "AXOS provides 57 functions, including audit trail and key-only file management functions, and 64 index paths vs. Btrieve's 31 operations and 24 access paths."
—PC TECH JOURNAL (Oct 1988)

McSoft, Inc.
P.O. Box 12044
Atlanta, GA 30355
404-984-0608 **RS# 230**

WHITNEY EDITOR $49

Small and fast. Uses all available memory. Split-screen editing. Configurable keyboard. Regular expression search. One-key compile. Features for writing documentation. Condensed/Outline display. Runs on IBM PC's, AT's, and PS/2's. USA shipping and handling $3; Outside USA $15. CA residents add sales tax.

Whitney Software, Inc.
P.O. Box 4999
Walnut Creek, CA 94596
415-933-9019 **RS# 298**

FREE TRIAL

Pinnacle File Manager

We guarantee immediate productivity gains from using our C file manager library. Simple table-and-column view of data with advanced B-Tree indexing and hi-speed cache frees your brain to solve application-specific problems.

Vermont Database Corp.
RR3 Box 6516
Stowe, VT 05672

802-253-4437

Number **328** on the Reader Service Card

sPORTt serial PORT tester

sPORTt is a serial port tester for people who don't want to be data communications experts, but use or write software for serial devices, such as plotters, mice, digitizers, modems, serial printers, or computer to computer links.

- **displays all data going in and out**
- **displays ASCII, decimal, hex, binary**
- **lets you choose handshaking lines**
- **sends/receives disk files**
- **automatically runs port diagnostics**

Order today only $89.95!

WHS
3037 Grass Valley Hwy #8201
Auburn, CA 95603

916-885-2480

Number **329** on the Reader Service Card

SHAREWARE AUTHORS

Give your programs the chance they deserve! With Megapost, your program is uploaded to CompuServe, GEnie, Delphi, BIX, and EXEC-PC The low fee is probably less than you would pay just to open accounts with all of these services. Included are one half-price update and notification of the downloading procedure and 3-month download count for each service. 15% discount for ASP members.

Write for a free brochure.

Andrew M. Saucci, Jr.
641 Koelbel Ct.
Baldwin, NY 11510-3915

Number **347** on the Reader Service Card

ADVERTISER INDEX

Reader Service	Advertiser Name	Page Number
305	Acme Software	61
282	Acucobol Inc.	81
345	Alphax Systems, Inc.	43
318	Andrews Data Systems	94
332	AtLast! Software Inc.	10
30	The Austin Code Works	87
207	Automation Education	61
51	Az-Tech Software, Inc.	94
343	Base Two Development	67
3	Borland International	C4
288	Burton Systems Software	94
321	Catenary Systems	27
335	CDP Consultants	95
*	CNS, Inc.	57
281	COBRA Systems Software	95
340	Computer Books	95
277	Covox Inc.	46
336	C Source	49
205	*C++ Report*	89
229	Disk Software Inc.	43
344	Duplication Tech Ltd.	96
316	EDI	C3
349	Eikon Systems	31
353	EMS	94
352	EZX	94
307	F. G. Soft Company	37
261	Genus Microprogramming	44
355	Geomaker Software	12
*	Gimpel Software	55
323	*Glenco Engineering Inc.*	17
334	Hamilton Laboratories	55
42	Harvard Softworks	46
313	High Aspect Development	95
348	Information Modes	96
294	Information Software Associates	94
317	Island Systems	21
226	Ithaca Street Software	27
346	JDyx Enterprises	43
295	Knowledge Dynamics Corporation	91
155	Komputerwerk Inc.	94
63	Laboratory Microsystems Incorporated	43
44	Lahey Computer Systems Inc.	6
246	Logical Alternatives Incorporated	94
271	Lugaru Software Ltd.	94
342	Library of Computer and Information Sciences	23
354	Maikon	95
223	Marietta Systems Inc.	94
291	MBAC	95
230	McSoft	96
350	*Microcomucopia*	91
232	Microsoft Human Resources	9
*	Microsoft Press	1
104	Microway	85
272	*Midnight Engineering*	95
121	MISOSYS, Inc.	82
266	Mix Software	59
94	MMC AD Systems	57
312	MTD Software Development Co. Ltd.	65
319	Mt. St. Helens Software	7
109	Nanosoft	65
278	Netwise	45
160	Nu-Mega Technologies	3
16	Opt-Tech Data Processing	96
*	Paradigm Systems	46
208	PC Tech, Inc.	37
6	The Periscope Company, Inc.	16
*	***PJ* Code DiskScription**	8
279	Pocket Soft, Inc.	11
*	***Programmer's Journal* Mail Order Store**	88
147	Quarterdeck Office Systems	C2
296	Que Corporation	30
324	Rogue Wave Software Associates	55
210	Ryle Design	95
26	Sage/Polytron Corporation	14
347	Saucci, Andrew M., Jr.	96
269	Sequiter Software Inc.	95
98	The Software Engineering Store	90
356	Software Projects Group	31
198	Software Science Inc.	13
221	Software Security, Inc.	5
110	Softway, Inc.	48
47	Solution Systems	70
10	South Mountain Software	35
330	Stanwood Associates Inc.	37
158	Summit Info Systems, Inc.	12
289	TechnoJock Software, Inc.	84
166	The Tool Makers	38
314	Tri-Star Distribution	95
276	Tri-Technology Systems Inc.	57
25	TurboPower Software	51
311	University Degree Advisory	94
125	V Communications, Inc.	71
341	V Communications, Inc.	59
328	Vermont Database Corporation	96
304	Visitech Software	27
320	Wenham Software Co.	95
298	Whitney Software, Inc.	96
339	W.H. Roetzheim	91
329	WHS	96
123	XenoSoft	95
168	Young Software Engineering	94

When contacting any of the above advertisers, please tell them you saw their ad in ***Programmer's Journal*** and that you appreciate their continuing support of this magazine. The advertisers with an * prefer to be contacted directly. ***PJ***'s Advertiser Index is provided as a free service, and ***Programmer's Journal*** cannot be held responsible for omissions or errors. ***Programmer's Journal*** is not responsible for errors in advertisements.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

A message from "the Author's Network"

You want high quality software, right? But you don't want to take a loan out to buy it. And you want it now. Then you should know about an exciting, new on-line service. It's called **EDI Author-Direct** and it's going to change the way you think about software.

EDI Author-Direct was created by software authors. On-line fees collected from users like yourself pay for the software you download. The result - you pay less for the software you download. How much less? In most cases you pay only for your on-line time, which may vary between 17 and 20 cents per minute (depending on the data network you choose to access us).

EDI Author-Direct is packed with great programs. Download a copy of **Sagewords**, J. Keith Phipps's word-processing program. It can be yours in minutes! Or how about William Halls' **Commo**, a fully developed communications program. Interested in protecting your computer from viruses? Get a copy of David Leithauser's **Virus Stopper II**. Need a mailing list program? Download **Label Master Version 4.0**. We use it ourselves. You name it, you can find it on EDI Author-Direct.

Since we are an author-direct network, you get the latest product releases direct from the author. If you have a problem, get back on-line and talk direct with the author. On-line forums put you in touch with authors so your questions can be answered quickly and accurately.

To get connected call one of the *Minitel Data Network* access numbers below and follow the instructions provided. If you cannot find a phone number in your immediate area, hook up with us via **BT TYMNET** (formally TYMNET), one of the nations premier data networks with local access numbers in over 850 cities and eighty countries worldwide. We will be on-line with BT TYMNET by mid March. For your local access number, call BT TYMNET customer support (24 hours a day) at 1-800-336-0149.

ACCESS INSTRUCTIONS (*Minitel Data Network*): Set your modem to 7-E-1 and call one of the numbers below. Once connected press RETURN until you see a # prompt. Then press **x** and RETURN to get a star prompt. Type .vtt (remember the period) to gain network access. Put in your credit card info to pay for the call and type EDI to access us on the network. On-line costs are 17 cents per minute, even at 2400 baud. There are no start-up costs. Prices subject to change without notice.

ALABAMA
Anniston (205)820-7609
Birmingham (205)254-3120
Montgomery (205)279-0314
ARIZONA
Phoenix (602)233-9592
ARKANSAS
Little Rock (501)376-1535
CALIFORNIA
Burbank (818)954-9161
Corona (714)736-8820
El Monte (818)442-9291
Garden Grove (714)895-4946
Los Angeles (213)772-6971
Monterey (408)899-7855
Mountain View (415)964-8240
Newport Beach (714)895-4946
No. Hollywood (818)786-4111
Sacramento (916)452-7356
San Bernardino (714)888-7611
San Diego (619)293-3620
San Fransisco (415)391-9080
San Jose (408)554-0344
Van Nuys (818)786-4111
COLORADO
Colorado Springs (719)540-8220
Denver (303)893-3803
CONNECTICUT
Hartford (203)728-6766
Milford (203)728-6766
New Haven (203)877-2751
Stamford (203)327-6745
Wilmington (302)571-1433
D.C.
Washington (301)937-0530
Washington (301)937-0722
FLORIDA
Fort Lauderdale (305)462-3104
Jacksonville (904)398-2074
Miami (305)651-2387
Pensacola (904)438-7203
Tampa (813)626-5235
GEORGIA
Atlanta (404)256-9151
Augusta (404)790-5406
Columbus (404)682-1103
HAWAII
Honolulu (808)524-7950
ILLINOIS
Chicago (312)704-0313
Oak Brook (312)572-8550
Oak Brook (312)572-8566
Rock Island (319)324-0942
Springfield (217)525-6683
INDIANA
Indianapolis (317)634-4301
IOWA
Davenport (319)324-0042
Des Moines (515)224-0011
KENTUCKY
Louisville (502)587-9560
LOUISIANA
New Orleans (504)566-0002
MARYLAND
Baltimore (301)760-1950
MASSACHUSETTS
Boston (617)965-4900
MICHIGAN
Detroit (313)963-9141
Grand Rapids (616)456-6047
MINNESOTA
Minneapolis (612)338-5540
MISSOURI
Kansas City (816)283-3719
St. Louis (314)721-4801
NEBRASKA
Omaha (402)341-2425
NEW JERSEY
Boonton (201)335-2332
Cherryhill (609)235-3220
Hackensack (201)489-9020
Moorestown (609)235-3220
New Brunswick (201)846-9310
Newark (201)564-6300
Parsippany (201)335-2332
Springfield (201)564-6300
Trenton (609)883-5113
Whippany (201)335-2332
NEW YORK
Buffalo (716)896-5410
Long Island (516)481-8028
New York (212)391-1920
New York (212)391-0370
Poughkeepsie (914)454-1808
Purchase (914)694-6842
Syracuse (315)428-1674
NORTH CAROLINA
Charlotte (704)374-0587
Fayetteville (919)436-0145
Raleigh (919)821-7535
NORTH DAKOTA
Fargo (701)237-5282
OHIO
Akron (216)535-6635
Cincinnati (513)241-1446
Cleveland (216)621-3880
Columbus (614)461-8450
OKLAHOMA
Lawton (405)355-0275
Oklahoma City (405)272-9751
OREGON
Portland (503)241-0549
PENNSYLVANIA
Harrisburg (717)236-5241
Philadelphia (215)829-0912
Pittsburgh (412)261-9305
SOUTH CAROLINA
Columbia (803)771-7665
SOUTH DAKOTA
Sioux Falls (605)334-1832
TENNESEE
Memphis (901)521-0529
Nashville (615)292-6801
TEXAS
Austin (512)499-8029
Dallas (214)638-7850
Fort Worth (817)460-3711
Houston (713)981-9797
Kileen (817)532-8980
San Antonio (512)525-1692
UTAH
Salt Lake (801)544-0086
VIRGINIA
Newport News (804)874-5121
Norfolk (804)858-2715
Richmond (804)788-0401
WASHINGTON
Seattle (206)624-3580
Spokane (509)455-9398
WEST VIRGINIA
Charleston (304)345-7505
Parkersburg (304)485-7569
WISCONSIN
Madison (608)255-5857

ELECTRIC DISTRIBUTION
INTERNATIONAL

Coming March 15th, BT TYMNET Access. Call 1-800-336-0149 for your local access number and type SOFTWARE at the login prompt.

1332 Hermosa Avenue, Suite 7, Hermosa Beach, CA 90254, (213) 374-3478, 800-676-6076

Number **316** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

BORLAND'S TURBO C PROFESSIONAL 2.0

CORPORATE TOP RATED PC WEEK 1989 SATISFACTION POLL

PC WEEK POLL: C COMPILERS

	Overall Weighted Score	Overall Reliability	Complete of Command Descript.	Overall Perform.	Complete & Organiz. Document.	Document Clarity	Compiling Process Efficiency	Product Support Quality	Value Relative To Cost	Product Support Access.
Turbo C 2.0 (Borland International)	81	87	79	84	77	78	86	72	70	93
C Optimizing Compiler 5.1 (Microsoft Corp.)	76	83	80	81	78	74	76	68	67	70
C++ 1.07 (Zortech Inc.)	66	68	64	71	63	63	69	60	58	76

"Microsoft was No. 1, but they have been unseated by Borland." PC Week, *May 8, 1989*

PC WEEK POLL: SOFTWARE DEBUGGERS

	Overall Weighted Score	Overall Reliability	Effective. Programmer Interface	Document. Clarity	Complete. Command Descript.	Complete. & Organize. Document	Overall Perform.	Integration Within Programming Environment	C Compiler Compatibility	Product Support Quality	Product Support Access	Value Relative To Cost
Turbo Debugger 1.0 (Borland International)	84	89	90	81	81	81	89	88	81	73	72	93
Codeview 2.2 (Microsoft Corp.)	73	80	71	72	74	74	74	74	78	67	64	72

"Borland's Debugger outshines Microsoft's Codeview." PC Week, *May 15, 1989*

It's two winners in one.

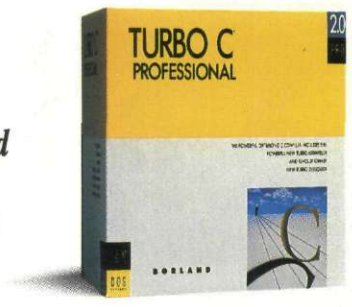
Turbo C,® the core of Turbo C Professional, was the outright winner in *PC Week*'s Poll of Corporate Satisfaction on C compilers. Overall, Borland won with 81. Microsoft® placed second.

Turbo Debugger,® also included in Turbo C Professional, was the outright winner in EVERY category in *PC Week*'s Poll Of Corporate Satisfaction on Debuggers. And, once again, we topped the score with 84, overall. Microsoft came in second-best, 11 points behind.

Get Borland's Turbo C Professional and get the best of both worlds: our top-rated C compiler and our top-rated Debugger.

Call **(800) 345-2888*** and we'll send you both *PC Week* polls and technical specifications on Turbo C and Turbo Debugger.

Turbo C Professional includes both Turbo C 2.0 and Turbo Assembler® & Debugger.



BORLAND

CODE: MC 53

From Canada, Call (408) 848-4391. Reprinted from PC Week, May 8, 1989; PC Week, May 15, 1989. Copyright ©1989 Ziff Communications Company. Turbo C, Turbo Debugger, and Turbo Assembler are registered trademarks of Borland International. Copyright ©1989 Borland International, Inc. All rights reserved. BI 1328

Number 3 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED